Trabajo de Fin de Grado

Grado en Ingeniería Civil

Búsqueda de Patrones Relacionados con la Obra Civil Derivados del uso de la RAP

Dep. Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Departamento de Ingeniería Gráfica

Trabajo de Fin de Grado
Grado en Ingeniería Civil

Búsqueda de Patrones Relacionados con la Obra Civil Derivados del uso de la RAP

Autor:
José Antonio Punta de la Herrán

Tutora:
Cristina Torrecillas Lozano
Profesora titular

Dep. de Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Carrera: Búsqueda de Patrones Relacionados con la Obra Civil Derivados del uso de la RAP

Autor: José Antonio Punta de la Herrán

Tutora: Cristina Torrecillas Lozano

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A A. y C.

Resumen

La actualización de la cartografía sigue siendo un desafío hoy en día. Sin embargo, herramientas como la Red Andaluza de Posicionamiento (RAP) pueden permitir avanzar en la materia, ya que los registros recopilados por la Red permiten identificar patrones al tratar los datos.

En este proyecto, la metodología seguida ha sido la creación de algoritmos escritos en lenguaje Python, mediante los cuales se trata de identificar las obras lineales que no estén reflejadas en el DERA. Esto se ha realizado en distintas fases. Una primera de extracción y simplificación de puntos que no fueran de interés (aislados y en núcleos urbanos). Después, se han identificado aquellas conexiones que ya estén reflejadas en el DERA, para poder diferenciar finalmente todas aquellas que no lo estén. Serán los patrones obtenidos de estas últimas las que permitirán realizar un análisis.

Finalmente se han obtenido una serie de resultados los cuales, aunque no demuestran una total eficacia de los algoritmos, sí que permiten atisbar un camino a seguir para poder obtener mejores resultados. También se analiza una posible forma de categorizar los patrones obtenidos mediante algoritmos de *Machine Learning*. A pesar de esto esta técnica no podrá ser llevada a cabo por la naturaleza de los datos, fuertemente dependiente del comportamiento aleatorio del topógrafo.

Abstract

Nowadays, the update of the cartography continues being a challenge. However, tools such as the Andalusian Network of Positioning (RAP) let us to improve in the matter, because the records of the Network allow us to identify patterns.

In this project, the methodology is based in creating algorithms in Python language. With these scripts we try to identify linear patterns that are not reflected in the DERA. This process has been done in different phases. Firstly, an extraction and simplification of points that are not useful (isolated or in urban zones). Afterwards, connections that were already reflected in the DERA have been identified, in order to differentiate from the "new" patterns.

Finally, we have obtained a series of results which, although they are not efficient enough, they let us to discover a way to keep researching. A possible way of categorizing patterns by using Machine Learning is also analyzed. Unfortunately, this technique cannot be carried out due to the nature of the data, highly dependent on the random behavior of the topographer.

Índice

Índice de Tablas

Índice de Figuras

Glosario

EPSG: European Petroleum Survey Group 13

ESRI: Environmental Systems Research Institute 8

ETRS89: European Terrestrial Reference System 1989 13

GNSS: Global Navigation Satellite System, 1

GPS: Global Positioning System 1

PNOA: Plan Nacional de Ortofotografía Aérea, 32

RINEX: Receiver INdependent EXchange 1

UTF-8: Formato de Transformación UCS/Unicode de 8 bits 14

UTM: Universal Transverse Mercator 13

WMS: Web Map Service, 32

Notación

A^*	Conjugado
c.t.p.	En casi todos los puntos
c.q.d.	Como queríamos demostrar
■	Como queríamos demostrar
e.o.c.	En cualquier otro caso
e	número e
IRe	Parte real
IIm	Parte imaginaria
sen	Función seno
tg	Función tangente
arctg	Función arco tangente
sen	Función seno
$\sin^x y$	Función seno de x elevado a y
$\cos^x y$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
rect	Función rectángulo
Sinc	Función sinc
∂y ∂x	Derivada parcial de y respecto
$x_\circ$	Notación de grado, x grados.
Pr(A)	Probabilidad del suceso A
SNR	Signal-to-noise ratio
MSE	Minimum square error
:	Tal que
<	Menor o igual
>	Mayor o igual
\	Backslash
⇔	Si y sólo si

1 INTRODUCCIÓN

La actualización cartográfica es un proceso costoso tanto en tiempo como en presupuesto. Hoy en día, en un mundo digital, la demanda de datos actualizados exige agilizar y focalizar esa actualización hacia los elementos que imponen más cambios en el terreno. La Red Andaluza de Posicionamiento poseé un servicio muy extendido para el uso de correcciones GPS que mejoran la precisión en la determinación de coordenadas. Mediante el registro de las posiciones de los usuarios que requieren el servicio se pueden identificar los trazados de algunas de las obras lineales ejecutadas tanto en el territorio andaluz como fuera de él.

Por otro lado, en este tema también se hablará de la herramiento geográfica elegida, que será en este caso el sistema de información geográfica denominado QGis. Se cree que es la opción más conveniente debido a que es un software de código abierto y que su uso está muy extendido incluso en la Junta de Andalucía.

También se tratará en este apartado del lenguaje de programación Python, pues será mediante el cual se automatizará la búsqueda de patrones lineales.

Por último, se indicarán los objetivos perseguidos por este trabajo.

1.1 La Red Andaluza de Posicionamiento

En el mundo de la construcción, la tendencia deriva hacia precisiones cada vez mayores. Para conseguir estas precisiones, en España, la solución suele pasar por el uso de estaciones permanentes. Estas estaciones suelen conformar redes de estaciones permanentes GNSS (*Global Navigation Satellite Systems*). En el despegue definitivo de este tipo de tecnología tuvo una importancia vital la expansión del uso de datos en internet vía móvil en los dispositivos GPS, ya que esto permitiría disponer de correcciones en tiempo real [1].

La Red Andaluza de Posicionamiento (RAP), ofrecida por la Junta de Andalucía, es una red GPS que permite obtener un posicionamiento preciso en todo el territorio andaluz. Esta gran precisión se consigue mediante la descarga gratuita de correcciones diferenciales y de archivos RINEX [2].

Las correcciones diferenciales serán aquellas que se darán en tiempo real (RTK) y la descarga de archivos RINEX irá asociada a la necesidad de conexión con un servidor que permita la recepción, el almacenamiento, el procesamiento y la comparación (post-proceso) de la información descargada [3].

La RAP fue desarrollada y es gestionada por el IECA (Instituto de Estadística y Cartografía de Andalucía). Este organismo sostiene una estructura formada por un sistema de control y 22 estaciones. Para garantizar la homogeneidad del servicio, se situará una estación por provincia, situando una adicional en Cádiz, concretamente en Algeciras, con visos de poder prestar un servicio de calidad en el estrecho de Gibraltar. Las

estaciones están situadas sobre edificios de carácter público que aseguren suministro eléctrico 24h al día, 7 días a la semana y 365 días al año, ver Figura 1-1 y Tabla 1-1.

Figura 1-1. Estaciones de la RAP

Tabla 1-1. Estaciones de la RAP

Ubicación	Código
Puerto Real	UCAD
Huelva	HULV
Algeciras	ALGC
Sevilla	SEVI
Málaga	MALG
Granada	GRAN
Almería	ALMR
Córdoba	CRDB
Jaén	UJAE
Ronda	ROND
Osuna	OSUN
Lebrija	LEBR
Aracena	ARAC
Pozoblanco	POZO
Calar Alto	CAAL
Las Viñas	ANDU
Villanueva del Arzobispo	VIAR
Castril	CAST
Cabra	CABR
Cazalla de la Sierra	CAZA
Motril	MOTR
Huércal Overa	HUOV

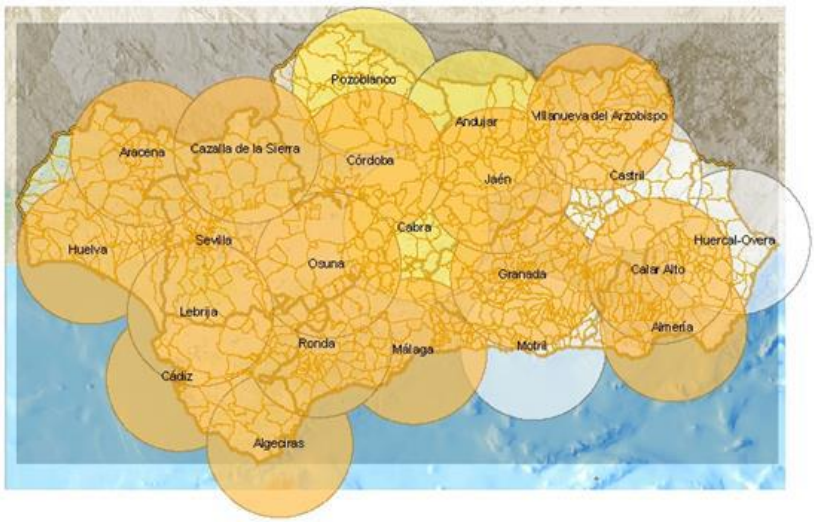


Figura 1-2. Cobertura RAP

Con la red descrita y representada en las imágenes, las estaciones no distarán más de 100 km una de otra, asegurándose así el correcto funcionamiento de la misma, ver Figura 1-2. Todas estas estaciones estarán gobernadas por el sistema de control. Este sistema estará formado básicamente por un servidor informático, con conexión a internet e intranet y con capacidad suficiente para manejar la red en su totalidad [1].

La red geodésica conformada por la RAP es coherente y compatible fuera del territorio andaluz. Esto quiere decir que el IECA proporciona un marco geodésico de referencia único y estable a nivel nacional e internacional.

El acceso a los datos es totalmente gratuito, pero para conseguirlo habrá que registrarse en el sistema. Para ello se introducirá un nombre de usuario y la compañía o institución a la que pertenece dicho usuario. Estos datos, junto con otros muchos (fecha, hora, posición …), serán los que queden en los registros que se utilizarán para la búsqueda de patrones lineales.

Por último, destacar el papel del Laboratorio de Astronomía, Geodesia y Cartografía de la Universidad de Cádiz como Centro de Control responsable del control geodésico de la red de estaciones, hasta el año 2018 que fue asumido por la empresa Leica Geosystems. El control es necesario por la actualización y ajuste de las coordenadas de las estaciones permanentes de la red, debido a las continuas modificaciones que van sufriendo. Estas modificaciones se irán produciendo según se vayan oficializando los Sistemas de Referencia Terrestres a nivel internacional, pero la falta de ajuste podría derivar en un mal funcionamiento de la red.

1.2 QGis

La representación del medio terrestre siempre ha sido una necesidad importante a la hora de desarrollas distintos softwares. El aumento en la cantidad de datos recopilados durante los proyectos o estudios que estén relacionados con el entorno ha desembocado en la aparición de programas específicos para el manejo, análisis, organización y modelización de datos geográficos. Hablamos ni más ni menos de los Sistemas de Información Geográfica (SIG). Un SIG no es necesariamente un programa informático, sino que es un conjunto de herramientas que aglutinen la capacidad de operar con datos geográficos.

Dentro del mundo de los programas dedicados a los SIG hay una gran variedad entre las que elegir. Para este trabajo, se decide utilizar el software de código abierto QGis. QGis, es un proyecto oficial de Open Source Geospatial Foundation (OSGeo) y es compatible con las plataformas más habituales (Windows, Mac y Linux) [4].

Figura 1-3. Versión QGis

La razón de esta elección es entre otras, que es el programa promovido por la Junta de Andalucía para el manejo de la información geográfica, aunque también se usen otras aplicaciones. Además de esto, el hecho de que sea un programa de código abierto licenciado bajo GNU (General Public License), permite el libre uso del programa, así como de todos sus algoritmos.

Otro aspecto importante es que se pueden desarrollar algoritmos, complementos o pequeñas aplicaciones en el propio programa. Esto se hará mediante scripts basados en código Python (del que se hablará más adelante), utilizando módulos propios del programa.

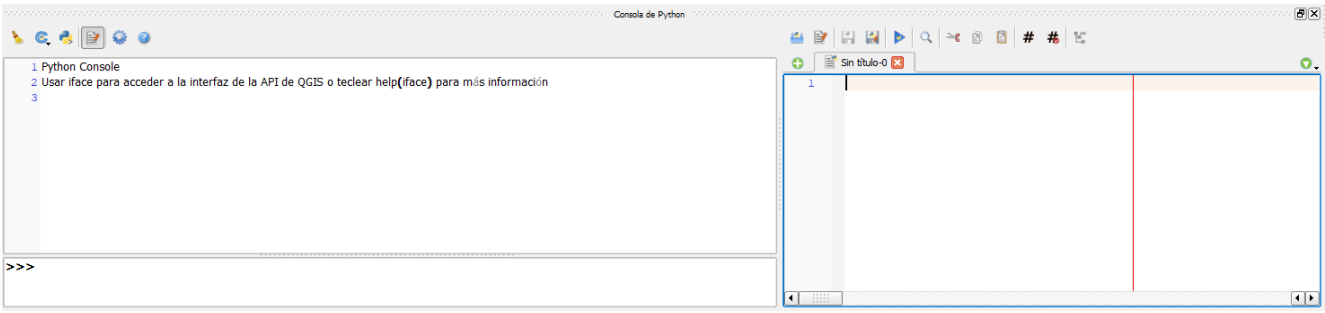


Figura 1-4. Consola de Python en QGis

QGis es un proyecto mantenido por voluntarios y organizaciones sin ánimo de lucro, por lo tanto, la forma de desarrollar nuevas funcionalidades, mantener actualizado el programa e informar y eliminar los posibles errores, se hará mediante la propia comunidad de QGIS. Vemos por consiguiente que QGis es un proyecto colaborativo La viabilidad del proyecto depende además de los desarrolladores, de donaciones y patrocinios..

Figura 1-5. Comunidad QGis

1.3 Python

Automatizar tareas mediante un código informático nos permite automatizar tareas de forma que ahorramos tiempo y esfuerzo en realizar dichas tareas. Además, esta metodología nos permite analizar muchos más datos más rápidamente, consiguiendo de tal manera una mayor capacidad de análisis.

Para programar las tareas existen muchísimos lenguajes, cada uno con sus ventajas e inconvenientes. En este trabajo emplearemos el código Python, por ser el que mejor se adecua a las particularidades de este trabajo. Como se dijo en el apartado anterior, Python, está integrado en QGis, lo cual será una gran ventaja.

Además, al igual que QGis, Python depende de una licencia de código abierto, concretamente la Python Software Foundation License (PSFL). Esta licencia tiene la particularidad de que permite modificaciones en su código fuente [5].

Figura 1-6. Logo Python

Otro aspecto muy interesante de Python son sus módulos, es decir agrupaciones de algoritmos creados por miembros de la comunidad de Python que pueden ser usados libremente. Además, estos módulos están escritos respetando unas ciertas "normas", lo cual los hace muy legibles e interpretables.

Al igual que el resto de lenguajes, Python tiene una serie de particularidades que lo identifican.

- Es un lenguaje orientado a objetos, en Python todo es un objeto.
- Es un lenguaje interpretado, es decir, el código se interpreta "línea a línea".
- Al ser un lenguaje interpretado, los erraros saltan en tiempo de ejecución.
- Está disponible para las principales plataformas (Windows, Linux, Mac).
- El tipado es dinámico, por tanto, el tipo de las variables puede cambiar según el caso.
- Es un lenguaje indentado, lo que hace fundamental mantener un orden en el uso del sangrado de las líneas.

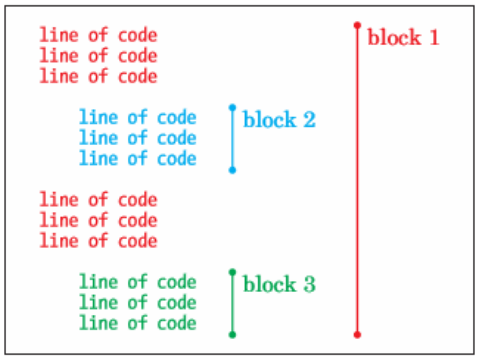


Figura 1-7. Lenguaje indentado

Python es un lenguaje que está ganando mucha popularidad a lo largo de los años, tendiendo un crecimiento y

expansión superior al resto de lenguajes. Tal es esta difusión que incluso ha llegado a ocupar el primer puesto del ranking IEEE Spectrum en 2017.

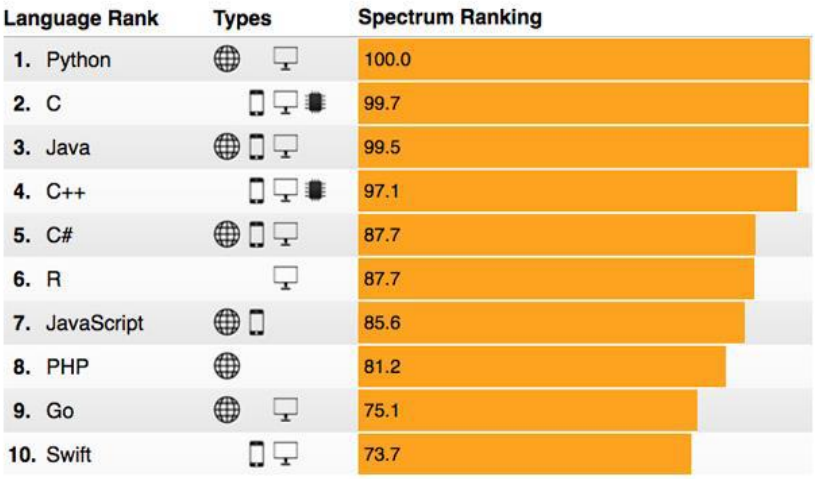


Figura 1-8. Ranking IEEE Spectrum 2017

Esta expansión es debida aparte de las ventajas ya comentadas anteriormente, a que se está convirtiendo en el lenguaje de referencia a la hora de programar algoritmos de *Machine Learning* [1]y *Deep Learning* [2]e incluso para el manejo de *Big Data*[3], campos de la computación con muchas expectativas de futuro.

Por último, cabe destacar que para programar en Python en el programa de SIG QGIS, habrá que respetar algunas pequeñas diferencias, pero la base será la misma. También habrá que importar los módulos de geoalgoritmos de QGis antes de utilizarlos.

1.4 Objetivo

Con este trabajo se pretende elaborar un proceso que permita identificar obras lineales en el territorio andaluz, a partir de las conexiones de la RAP, de forma automática a partir de un algoritmo informático. Su validez será corroborada mediante la comparación entre las obras encontradas mediante esa metodología con las que oficialmente constan, por ejemplo, en los Datos Espaciales de Referencia de Andalucía (*DERA*).

Con el procedimiento establecido, se podría utilizar para las conexiones de cada año de forma que se identificarán de forma directa las variantes construidas durante ese periodo, o las modificaciones que no han sido bien reflejadas. De esta forma se identifican anualmente los cambios en el terreno, y se puede ayudar a decidir qué zonas presentan más o menos cambios en una futura actualización de cartografía.

La automatización del proyecto mediante programación (en código Python), además de hacer del mismo una herramienta válida para las conexiones de los años siguientes, abre la posibilidad de una futurible aplicación con la que la Junta de Andalucía podría tener el control cartográfico de gran parte de las obras lineales construidas en Andalucía, así como de sus modificaciones.

[1] Tecnología de programación basada en algoritmos con reglas que permitan crear otras nuevas a medida que se va desarrollando.
[2] Tecnología que lleva el *Machine Learning* a un nivel superior creando un sistema por capas o unidades neuronales, intentando asemejarse al funcionamiento del cerebro humano.
[3] Se refiere a un volumen de datos tal, que dificuta o incluso impide su manejo mediante las herramientas convencionales.

2 DATOS DE PARTIDA

En este apartado se abordará los diferentes aspectos relativos a los datos de partida, desde su obtención hasta su tratamiento. Se mencionarán tanto las conexiones de la RAP, como los distintos datos de los núcleos urbanos, carreteras o, por ejemplo, ferrocarriles. Cada uno tendrá un origen distinto y, por tanto, diferentes particularidades que habrá que tener en cuenta.

2.1 Conexiones RAP

Como se explicó anteriormente, el uso de la Red Andaluza de Posicionamiento para correcciones vía GPS deja un registro en aquella en las que el usuario prefiere una corrección con solución de la red. A lo largo del año, el organismo andaluz recopila todas las conexiones de los diferentes usuarios que han usado en algún momento la red, junto con su posición y diferentes datos de interés (fecha, hora, empresa, etc.). Este tipo de información se almacena en archivos de registros .log para cada mes. Con estos archivos, se puede establecer una base de datos para las conexiones de cada año.

Las conexiones almacenadas como bases de datos se pueden exportar como archivos de texto separados por comas (CSV). Esta exportación se puede hacer con alguno de los programas incluidos en el paquete *Microsoft Office*, decantándonos en este caso por la herramienta *MS Excel.*

Estos archivos en formato CSV se pueden importar en el visor de QGis, tanto como tabla como capa geográfica si de ello se tratara. En nuestro caso, se importará como capa de puntos, puesto que dos de los campos de las conexiones representan las coordenadas necesarias para su representación.

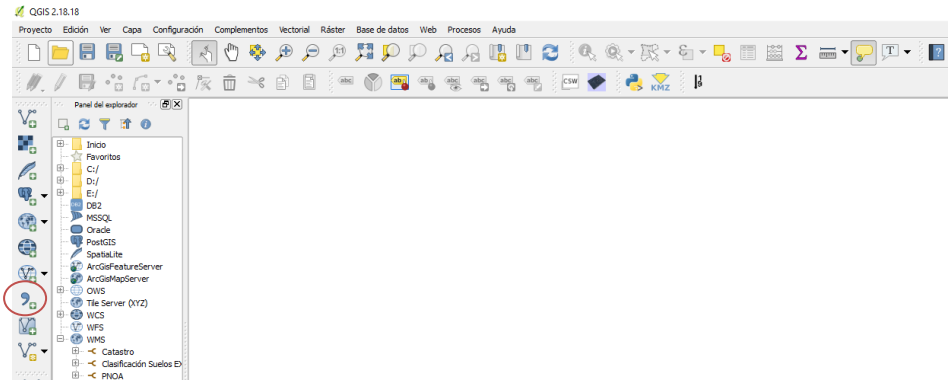


Figura 2-1. Importar CSV en QGis

Una vez importadas las conexiones se muestran en el visor como se ve en la siguiente imagen. Se puede apreciar en el borde inferior izquierdo que las conexiones de cada año serán representadas por capas distintas. Al estar representadas en QGis, se pueden guardar como un archivo en formato shapefile de ESRI.

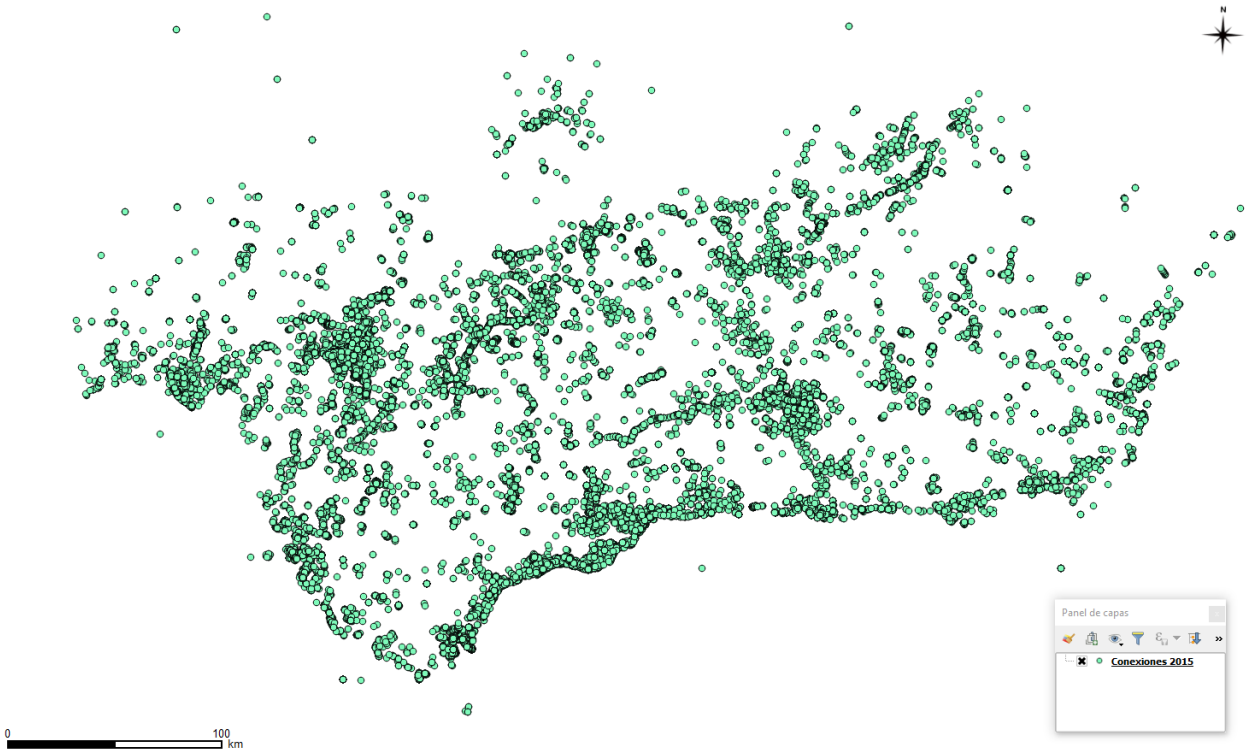


Figura 2-2. Capa de conexiones en QGis

2.1.1 Campos

Para todas las capas correspondientes a las conexiones de cada uno de los años, se mantendrán los mismos campos de datos. Esto será así principalmente por dos motivos:

- Se ganará claridad en los datos, siendo más fácil organizarlos, ordenarlos, relacionarlos y referenciarlos.
- Se podrán utilizar los mismos algoritmos para todas las capas, puesto que los campos siempre estarán en la misma posición.

Tabla 2-1. Campos de atributos

Nº	Campo	Descripción
1	Id	Identificador de orden de cada uno de los puntos
2	User/UserName	Nombre del usuario que utilizó el servicio
3	Company	Empresa o institución a la que pertenece el usuario que utilizó el servicio
4	StartDate	Fecha en la que se inicia la conexión al servicio
5	StartTime	Hora en la que se inicia la conexión al servicio
6	EndDate	Fecha en la que finaliza la conexión al servicio
7	EndTime	Hora en la que finaliza la conexión al servicio
8	Duration	Diferencia de tiempo entre el inicio y el fin de la conexión
9	Product/ProductName	Red de la que se obtendrán las correcciones de la RAP
10	Lat	Latitud correspondiente a las coordenadas geográficas ETRS89
11	Long	Longitud correspondiente a las coordenadas geográficas ETRS89
12	Height	Cota z del punto

En la tabla 2-1, se muestran los diferentes campos de datos presentes en todas las capas de conexiones.

Cabe destacar que debido operaciones posteriores necesarias para el tratamiento de los datos será necesaria la creación de los siguientes campos:

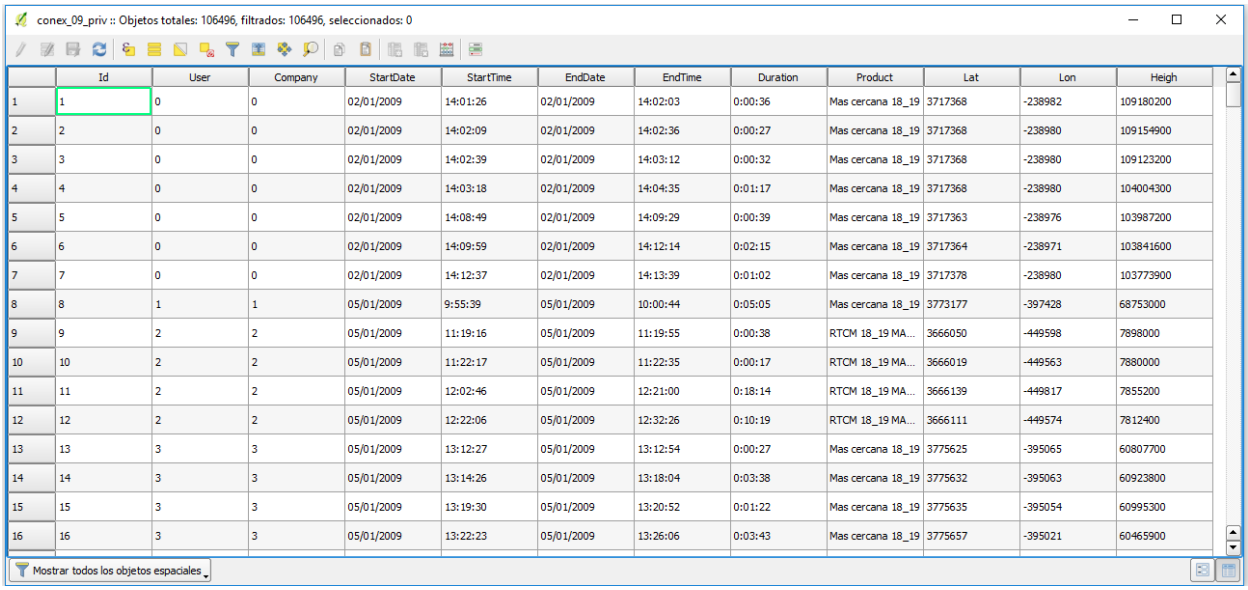
- 13 SRC_X: coordenada x correspondiente a la proyección UTM ETRS89 Huso 30.
- 14 SRC_Y: coordenada y correspondiente a la proyección UTM ETRS89 Huso 30.
- 15 Grupo: índice que indicará la posible pertenencia a una misma obra de un conjunto de datos.

2.1.2 Privacidad

De los diferentes campos de datos presentes en las conexiones anuales de la RAP, habrá algunos especialmente sensibles por su posible intromisión en la privacidad del usuario del servicio. Se tratará concretamente del campo que contiene el propio nombre del usuario (*User* o *UserName*) y del campo que contiene el nombre de la empresa o institución a la que pertenece el usuario (*Company*).

Para poder operar con los datos sin perder la privacidad de los mismos, se asignará automáticamente un valor numérico a cada usuario diferente, teniendo el mismo valor los usuarios que presenten el mismo nombre. Se procederá de la misma manera con el campo Company. De esta forma conseguimos presentar las conexiones como una serie de Datos Anonimizados.

Las tablas de atributos de cada una de las capas presentarán la apariencia mostrada en la figura 2-3:



	Id	User	Company	StartDate	StartTime	EndDate	EndTime	Duration	Product	Lat	Lon	Heigh
1	1	0	0	02/01/2009	14:01:26	02/01/2009	14:02:03	0:00:36	Mas cercana 18_19	3717368	-238982	109180200
2	2	0	0	02/01/2009	14:02:09	02/01/2009	14:02:36	0:00:27	Mas cercana 18_19	3717368	-238980	109154900
3	3	0	0	02/01/2009	14:02:39	02/01/2009	14:03:12	0:00:32	Mas cercana 18_19	3717368	-238980	109123200
4	4	0	0	02/01/2009	14:03:18	02/01/2009	14:04:35	0:01:17	Mas cercana 18_19	3717368	-238980	104004300
5	5	0	0	02/01/2009	14:08:49	02/01/2009	14:09:29	0:00:39	Mas cercana 18_19	3717363	-238976	103987200
6	6	0	0	02/01/2009	14:09:59	02/01/2009	14:12:14	0:02:15	Mas cercana 18_19	3717364	-238971	103841600
7	7	0	0	02/01/2009	14:12:37	02/01/2009	14:13:39	0:01:02	Mas cercana 18_19	3717378	-238980	103773900
8	8	1	1	05/01/2009	9:55:39	05/01/2009	10:00:44	0:05:05	Mas cercana 18_19	3773177	-397428	68753000
9	9	2	2	05/01/2009	11:19:16	05/01/2009	11:19:55	0:00:38	RTCM 18_19 MA...	3666050	-449598	7898000
10	10	2	2	05/01/2009	11:22:17	05/01/2009	11:22:35	0:00:17	RTCM 18_19 MA...	3666019	-449563	7880000
11	11	2	2	05/01/2009	12:02:46	05/01/2009	12:21:00	0:18:14	RTCM 18_19 MA...	3666139	-449817	7855200
12	12	2	2	05/01/2009	12:22:06	05/01/2009	12:32:26	0:10:19	RTCM 18_19 MA...	3666111	-449574	7812400
13	13	3	3	05/01/2009	13:12:27	05/01/2009	13:12:54	0:00:27	Mas cercana 18_19	3775625	-395065	60807700
14	14	3	3	05/01/2009	13:14:26	05/01/2009	13:18:04	0:03:38	Mas cercana 18_19	3775632	-395063	60923800
15	15	3	3	05/01/2009	13:19:30	05/01/2009	13:20:52	0:01:22	Mas cercana 18_19	3775635	-395054	60995300
16	16	3	3	05/01/2009	13:22:23	05/01/2009	13:26:06	0:03:43	Mas cercana 18_19	3775657	-395021	60465900

Figura 2-3. Campos anonimizados

2.1.3 Ordenación de conexiones

Para el correcto funcionamiento de los algoritmos de identificación de obras lineales, será fundamental que los puntos con los que vamos a tratar tuvieran el orden adecuado para poder relacionarlos.

La forma de recorrer las capas en QGis será mediante un iterador que seguirá el orden de un identificador interno de las capas. Normalmente las capas suelen venir ordenadas temporalmente, pero debido al sucesivo guardado de las capas este orden puede desvanecerse.

Siempre que se dé este caso, las capas deben ser reordenadas siguiendo criterios lógicos. Aprovechando esta situación, se optará por reordenar todas las capas de forma que se optimice la posterior identificación de obras lineales.

Para poder hacerlo, se importarán los archivos .log de los registros de las conexiones en Excel. Una vez realizadas las modificaciones pertinentes para conseguir los mismos campos de datos en las conexiones de todos los años, se procede a ordenar los datos.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Id	User	Company	StartDate	StartTime	EndDate	EndTime	Duration	Product	Lat	Lon	Heigh
2	1	0	0	02/01/2009	14:01:26	02/01/2009	14:02:03	0:00:36	Mas cercana 18_19	3717368	-238982	109180200
3	2	0	0	02/01/2009	14:02:09	02/01/2009	14:02:36	0:00:27	Mas cercana 18_19	3717368	-238980	109154900
4	3	0	0	02/01/2009	14:02:39	02/01/2009	14:03:12	0:00:32	Mas cercana 18_19	3717368	-238980	109123200
5	4	0	0	02/01/2009	14:03:18	02/01/2009	14:04:35	0:01:17	Mas cercana 18_19	3717368	-238980	104004300
6	5	0	0	02/01/2009	14:08:49	02/01/2009	14:09:29	0:00:39	Mas cercana 18_19	3717363	-238976	103987200
7	6	0	0	02/01/2009	14:09:59	02/01/2009	14:12:14	0:02:15	Mas cercana 18_19	3717364	-238971	103841600
8	7	0	0	02/01/2009	14:12:37	02/01/2009	14:13:39	0:01:02	Mas cercana 18_19	3717378	-238980	103773900
9	8	1	1	05/01/2009	9:55:39	05/01/2009	10:00:44	0:05:05	Mas cercana 18_19	3773177	-397428	68753000
10	9	2	2	05/01/2009	11:19:16	05/01/2009	11:19:55	0:00:38	RTCM 18_19 MALAGA	3666050	-449598	7898000
11	10	2	2	05/01/2009	11:22:17	05/01/2009	11:22:35	0:00:17	RTCM 18_19 MALAGA	3666019	-449563	7880000
12	11	2	2	05/01/2009	12:02:46	05/01/2009	12:21:00	0:18:14	RTCM 18_19 MALAGA	3666139	-449817	7855200
13	12	2	2	05/01/2009	12:22:06	05/01/2009	12:32:26	0:10:19	RTCM 18_19 MALAGA	3666111	-449574	7812400
14	13	3	3	05/01/2009	13:12:27	05/01/2009	13:12:54	0:00:27	Mas cercana 18_19	3775625	-395065	60807700
15	14	3	3	05/01/2009	13:14:26	05/01/2009	13:18:04	0:03:38	Mas cercana 18_19	3775632	-395063	60923800
16	15	3	3	05/01/2009	13:19:30	05/01/2009	13:20:52	0:01:22	Mas cercana 18_19	3775635	-395054	60995300
17	16	3	3	05/01/2009	13:22:23	05/01/2009	13:26:06	0:03:43	Mas cercana 18_19	3775657	-395021	60465900
18	17	3	3	05/01/2009	13:27:46	05/01/2009	13:38:26	0:10:39	Mas cercana 18_19	3775659	-395018	60447400
19	18	3	3	05/01/2009	13:39:54	05/01/2009	13:41:16	0:01:21	Mas cercana 18_19	3775608	-395042	61185700
20	19	4	4	07/01/2009	8:20:19	07/01/2009	8:21:36	0:01:17	RTCM 20_21 MALAGA	3667411	-454996	3716100
21	20	4	4	07/01/2009	8:23:01	07/01/2009	8:23:41	0:00:40	RTCM 20_21 MALAGA	3667411	-454995	3911000
22	21	4	4	07/01/2009	8:24:55	07/01/2009	8:25:50	0:00:54	RTCM 20_21 MALAGA	3667411	-454995	3911400
23	22	3	3	07/01/2009	8:36:48	07/01/2009	8:41:48	0:04:59	Mas cercana 18_19	3710096	-436673	73348100
24	23	3	3	07/01/2009	8:43:06	07/01/2009	8:44:00	0:00:54	Mas cercana 18_19	3710095	-436673	73332200
25	24	3	3	07/01/2009	8:46:01	07/01/2009	8:49:05	0:03:03	Mas cercana 18_19	3710095	-436672	73265900
26	25	3	3	07/01/2009	8:50:15	07/01/2009	8:52:05	0:01:49	Mas cercana 18_19	3710095	-436672	73286900
27	26	5	5	07/01/2009	8:50:42	07/01/2009	8:52:24	0:01:41	Mas cercana 18_19	3714557	-686067	888700
28	27	3	3	07/01/2009	9:18:05	07/01/2009	9:20:35	0:02:29	Mas cercana 18_19	3713868	-429880	72827300
29	28	3	3	07/01/2009	9:21:47	07/01/2009	9:22:31	0:00:43	Mas cercana 18_19	3713868	-429881	72512900
30	29	3	3	07/01/2009	9:23:46	07/01/2009	9:25:44	0:01:58	Mas cercana 18_19	3713868	-429881	72511900
31	30	3	3	07/01/2009	9:26:59	07/01/2009	9:29:15	0:02:15	Mas cercana 18_19	3713868	-429881	72436900
32	31	3	3	07/01/2009	9:30:40	07/01/2009	9:32:41	0:02:00	Mas cercana 18_19	3713867	-429881	72440400
33	32	3	3	07/01/2009	9:33:53	07/01/2009	9:35:52	0:01:58	Mas cercana 18_19	3713867	-429881	72449400
34	33	6	6	07/01/2009	9:29:11	07/01/2009	9:37:43	0:08:32	Mas cercana 20_21	3689442	-610027	858000
35	34	3	3	07/01/2009	9:37:08	07/01/2009	9:38:19	0:01:11	Mas cercana 18_19	3713868	-429882	72362100
36	35	6	6	07/01/2009	9:37:45	07/01/2009	9:43:43	0:05:57	Mas cercana 20_21	3689173	-610076	1284000
37	36	6	6	07/01/2009	9:43:44	07/01/2009	9:44:18	0:00:33	Mas cercana 20_21	3689155	-610078	1371700
38	37	7	7	07/01/2009	9:35:57	07/01/2009	9:46:58	0:11:01	Mas cercana 18_19	3803581	-410497	19934800

Figura 2-4. Datos importados en Excel

Mediante herramientas de filtro de Excel se establece un criterio de ordenación que facilite la tarea al algoritmo de identificación de obras lineales.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Id	User	Company	StartDate	StartTime	EndDate	EndTime	Duration	Product	Lat	Lon	Heigh
2	1	0	0	02/01/2009	14:01:26	02/01/2009	14:02:03	0:00:36	Mas cercana 18_19	3717368	-238982	109180200
3	2	0	0	02/01/2009	14:02:09	02/01/2009	14:02:36	0:00:27	Mas cercana 18_19	3717368	-238980	109154900
4	3	0	0	02/01/2009	14:02:39	02/01/2009	14:03:12	0:00:32	Mas cercana 18_19	3717368	-238980	109123200
5	4	0	0	02/01/2009	14:03:18	02/01/2009	14:04:35	0:01:17	Mas cercana 18_19	3717368	-238980	104004300
6	5	0	0	02/01/2009	14:08:49	02/01/2009	14:09:29	0:00:39	Mas cercana 18_19	3717363	-238976	103987200
7	6	0	0	02/01/2009	14:09:59	02/01/2009	14:12:14	0:02:15	Mas cercana 18_19	3717364	-238971	103841600
8	7	0	0	02/01/2009	14:12:37	02/01/2009	14:13:39	0:01:02	Mas cercana 18_19	3717378	-238980	103773900
9	5392	0	0	23/01/2009	9:52:09	23/01/2009	9:53:11	0:01:01	Mas cercana 18_19	3682748	-244061	6019800
10	5393	0	0	23/01/2009	9:53:12	23/01/2009	9:53:46	0:00:33	Mas cercana 18_19	3682748	-244061	6030400
11	5395	0	0	23/01/2009	9:57:44	23/01/2009	10:00:09	0:02:25	Mas cercana 18_19	3682747	-244058	6029800
12	5465	0	0	23/01/2009	11:28:36	23/01/2009	11:46:14	0:17:38	Mas cercana 18_19	3680506	-207779	7855300
13	5472	0	0	23/01/2009	11:48:44	23/01/2009	11:58:13	0:09:29	Mas cercana 18_19	3680598	-207645	8643000
14	5475	0	0	23/01/2009	11:58:15	23/01/2009	12:02:18	0:04:03	Mas cercana 18_19	3680611	-207730	8821900
15	5477	0	0	23/01/2009	12:03:47	23/01/2009	12:06:51	0:03:03	Mas cercana 18_19	3680621	-207793	8803900
16	5482	0	0	23/01/2009	12:07:57	23/01/2009	12:13:12	0:05:14	Mas cercana 18_19	3680602	-207702	8990300
17	5483	0	0	23/01/2009	12:13:17	23/01/2009	12:14:49	0:01:31	Mas cercana 18_19	3680594	-207714	8829500
18	6631	0	0	27/01/2009	15:26:03	27/01/2009	15:43:19	0:17:16	Mas cercana 18_19	3692324	-246055	13371900
19	6634	0	0	27/01/2009	15:43:23	27/01/2009	15:44:52	0:01:29	Mas cercana 18_19	3692324	-246055	13528400
20	6641	0	0	27/01/2009	15:46:37	27/01/2009	16:07:02	0:20:24	Mas cercana 18_19	3692355	-246037	13197000
21	6649	0	0	27/01/2009	16:07:03	27/01/2009	16:12:24	0:05:21	Mas cercana 18_19	3692328	-246040	13403500
22	6704	0	0	27/01/2009	16:12:28	27/01/2009	16:46:15	0:33:47	Mas cercana 18_19	3692319	-246060	13418100
23	10445	0	0	14/02/2009	12:08:34	14/02/2009	12:15:40	0:07:05	Mas cercana 18_19	3740100	-222034	56346500
24	12071	0	0	19/02/2009	13:55:04	19/02/2009	13:56:18	0:01:14	Mas cercana 18_19	3746001	-383808	91558000
25	12078	0	0	19/02/2009	13:56:43	19/02/2009	13:59:42	0:02:59	Mas cercana 18_19	3745994	-383805	91342100
26	12080	0	0	19/02/2009	13:59:46	19/02/2009	14:00:24	0:00:38	Mas cercana 18_19	3745993	-383805	91605800
27	12090	0	0	19/02/2009	14:00:26	19/02/2009	14:18:45	0:18:19	Mas cercana 18_19	3746001	-383808	91378700
28	12092	0	0	19/02/2009	14:18:47	19/02/2009	14:19:57	0:01:09	Mas cercana 18_19	3746001	-383808	91947400
29	12095	0	0	19/02/2009	14:21:31	19/02/2009	14:24:26	0:02:54	Mas cercana 18_19	3746254	-383919	92106400
30	12104	0	0	19/02/2009	14:31:16	19/02/2009	14:37:06	0:05:49	Mas cercana 18_19	3745457	-383685	92861900
31	12153	0	0	19/02/2009	16:35:03	19/02/2009	16:36:02	0:00:58	Mas cercana 18_19	3743915	-384571	87877000
32	12193	0	0	19/02/2009	17:34:16	19/02/2009	17:34:48	0:00:31	Mas cercana 18_19	3746252	-383642	93306500
33	12194	0	0	19/02/2009	17:34:51	19/02/2009	17:35:40	0:00:48	Mas cercana 18_19	3746252	-383642	93309900
34	12195	0	0	19/02/2009	17:35:52	19/02/2009	17:36:26	0:00:33	Mas cercana 18_19	3746252	-383641	93329800
35	12196	0	0	19/02/2009	17:36:32	19/02/2009	17:37:36	0:01:04	Mas cercana 18_19	3746252	-383642	93346400
36	12197	0	0	19/02/2009	17:37:40	19/02/2009	17:38:09	0:00:28	Mas cercana 18_19	3746253	-383640	93138100
37	12204	0	0	19/02/2009	17:38:23	19/02/2009	17:51:21	0:12:57	Mas cercana 18_19	3747960	-384030	96814300
38	12247	0	0	19/02/2009	18:53:36	19/02/2009	19:01:45	0:08:09	Mas cercana 18_19	3745010	-385181	105797700

Figura 2-5. Datos ordenados

Se elige, en este caso, establecer el siguiente criterio:

- Ordenar en primer lugar por usuario.
- Ordenar después por fecha de fin de la conexión.
- Por último, ordenar por la hora a la que termina la conexión.

Las razones de este criterio, por la que además de mantener un orden se simplifican las tareas de identificación serás las siguientes:

- Los puntos quedarán agrupados por usuario, por lo que la comparación entre puntos sucesivos verificará casi siempre este primer criterio (no será así cuando se llegue a un cambio de un usuario a otro).
- Puntos con el mismo usuario quedarán ordenados por fecha, por lo que quedarán agrupados los puntos pertenecientes a días iguales.
- Los puntos correspondientes a fechas iguales quedarán ordenados por hora del fin de la conexión, de forma que los puntos serán recorridos posteriormente por el iterador en el orden en que el usuario los tomó.

2.2 DERA

Para la obtención de algunas capas fundamentales tanto para la limpieza, como posteriormente para la comprobación de la existencia de ciertas obras lineales, se utilizarán los Datos Espaciales de Referencia de Andalucía. Estos datos pertenecen al Instituto de Estadística y Cartografía de Andalucía, organismo asociado a la Conserjería de Economía y Conocimiento de la Junta de Andalucía.

Todos los datos se encuentran en formato shapefile (.shp), incluyendo además un PDF de metadatos, donde se aporta la siguiente información:

- Información de los datos: propósito, limitación de uso, mantenimiento, etc.
- Calidad de los datos: descripción de las fuentes y pasos de proceso.
- Representación espacial: tipología de cada capa (polígono, ráster, punto).
- Sistema de referencia terrestre
- Codificación de caracteres
- Información del contenido: nombre de los distintos objetos, breve descripción y fechas de referencia.
- Información sobre la referencia: contacto y opciones de transferencia digital.
- Información de los metadatos: organización de la que dependen los datos.

Para descargar los datos únicamente hay que elegir entre una de las 20 categorías posibles, y acceder a ella. En cada uno de los grupos se describe la información de las capas que vienen incluidas. Una vez encontrados los datos buscados se obtienen por descarga directa en la propia web.

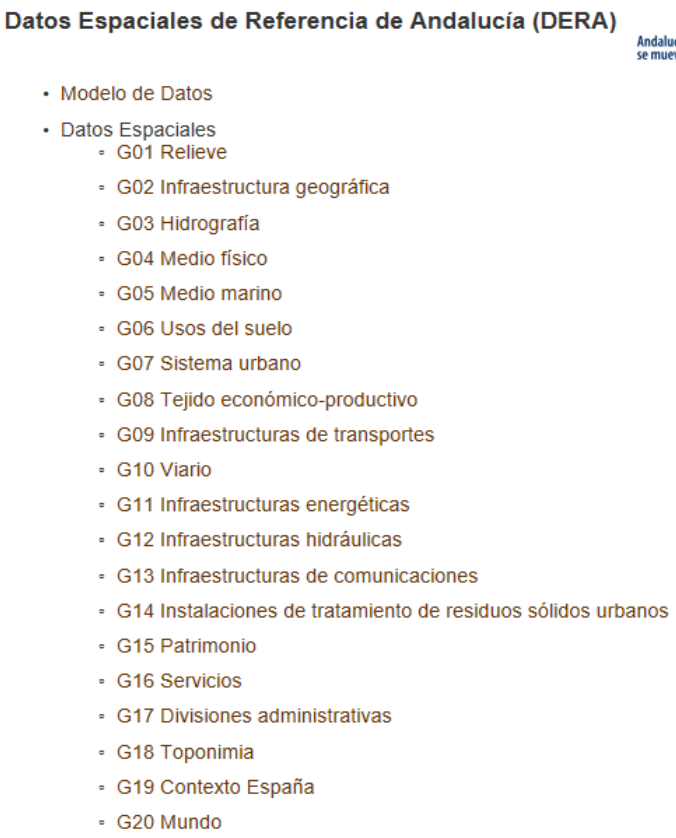


Figura 2-6. Categorías del DERA

Habrá que prestar especial atención a la fecha de actualización de cada una de las capas, puesto que cada una de ellas tendrá, en general, una fecha diferente, difiriendo en varios años. En la web del DERA existe un Registro de Actualizaciones en el que ver de forma compacta la fecha de actualización de todas las capas presentes.

Datos Espaciales de Referencia de Andalucía (DERA)

Registro de actualizaciones

Grupos y capas	Fecha de actualización
G01 Relieve	
Altimetría (rl03_altimetria)	05/11/2014
Intervalos altimétricos (rl04_inter_altimetria)	05/11/2014
Batimetría (rl05_batimetría)	05/11/2014
Intervalos batimétricos (rl06_inter_batimetria)	05/11/2014
Modelo Digital del Terreno (rl07_mdt)	05/11/2014
G02 Infraestructura geográfica	
Localización de las estaciones de la Red Andaluza de Posicionamiento (ig02_RAP)	10/02/2016
Líneas límite de los términos municipales (ig04_lindero) - Eliminada	03/04/2018
Mojones de líneas límite (ig05_mojon) - Eliminada	03/04/2018
Cuadrícula cartográfica 50.000 (ig07_cuadricula_50)	19/12/2014
Malla estadística 250x250 (ig08_malla_250m)	22/12/2017
Malla estadística 1Kmx1Km (ig09_malla_1Km)	22/12/2017
G03 Hidrografía	
Cursos fluviales(hd01_1_rio)	21/02/2014
Rutas (hd01_2_rio_ruta)	21/02/2014
Láminas de agua (hd02_lamina)	21/02/2014
Cuencas y subcuencas hidrográficas (hd03_cuenca)	05/11/2014
Fuentes y manantiales (hd04_manantial)	28/04/2017

Figura 2-7. Registro de actualizaciones del DERA

Para este proyecto, se requerirán algunas capas para el filtrado previo de los datos y otras para las comprobaciones de obra nueva.

2.3 Sistema de Referencia Terrestre

Siempre que tratemos con datos geográficos será de especial importancia la elección y el correcto uso de los diferentes sistemas de referencia. Para el correcto uso de los datos, lo más adecuado será utilizar el mismo sistema de referencia para todas las capas que utilicemos, independientemente de su sistema de referencia original.

En este trabajo, se utiliza un sistema de coordenadas proyectadas (x,y) oficial en España. Concretamente, el sistema EPSG: 25830 (Proyección UTM de ETRS89).

Los sistemas de referencia originales de cada uno de los diferentes datos son los siguientes:

- Conexiones Anuales RAP: sus coordenadas serán geográficas elipsoidales, correspondientes al sistema de referencia EPSG: 4258 (ETRS89). Por tanto, se importarán estos datos en su sistema original y posteriormente será necesario guardarlos en el sistema común elegido, procediendo de esta forma a su proyección cartográfica.
- Capas y datos provenientes del DERA: todas sus capas estarán representadas mediante coordenadas UTM correspondientes al EPSG: 25830, por lo que no será necesaria ninguna transformación adicional.

2.4 Codificación

La codificación de caracteres consiste en el establecimiento de un código mediante el cual asignamos un símbolo interpretable informáticamente a cada carácter propio de un lenguaje escrito. Es decir, permite interpretar y representar un alfabeto en un ordenador.

Hay diferentes formatos de codificación lo cual puede dar lugar a fallos o malas interpretaciones de algunos caracteres. Un error muy habitual es que las letras acentuadas se “conviertan” en otros símbolos totalmente distintos.

Para evitar estos errores en la medida de lo posible todos los datos, ya sean archivos de datos separados por comas (CSV), capas en formato shapefile, etc., serán representados en formato de codificación UTF-8.

Este formato presentará como mayor ventaja que con él se podrá representar cualquier carácter Unicode, estándar en el que están presentes todos los símbolos del castellano (acentos, letra ñ, etc.).

3 METODOLOGÍA SEGUIDA

La metodología a seguir en este trabajo estará orientada a conseguir relacionar las conexiones de partida con posibles actuaciones o variantes en obras lineales. Para trabajar de la forma más eficiente posible, previamente a la búsqueda de obras lineales, habrá que reducir el volumen de datos de entrada.

Después se procederá a la identificación de obras lineales propiamente dicha, utilizando algún algoritmo creado para tal fin.

El esquema de la metodología empleada queda representado por el siguiente gráfico.

Figura 3-1. Metodología aplicada

3.1 Prefiltrado de los Datos

Una primera operación que habrá que realizar para facilitar el posterior manejo de las conexiones, y ahorrar tiempo de computación, será reducir los datos de partida, eliminando los que de antemano se sepa que no serán útiles para el análisis.

3.1.1 Filtrado de Conexiones en Zona Urbana

Las obras lineales de gran calado (carreteras, líneas eléctricas, ferrocarriles, etc.) suelen estar asociadas a zonas interurbanas. Además, aunque existieran actuaciones en núcleos poblacionales, estas precisarían de una mayor profundidad de análisis y precisión con respecto a la que se puede conseguir con este tipo de estudio. A esto se une la dificultad de separar unas actuaciones de otras en zona urbana debido al solapamiento y acumulación de conexiones a lo largo de año.

Para delimitar los núcleos de población se utilizará la capa poligonal de núcleos urbanos del DERA denominada Su01 (30-11-2016). En esta capa están contenidos todos los núcleos urbanos de Andalucía junto con sus límites territoriales en formato Shapefile, de forma que cada polígono se corresponderá con todo el territorio perteneciente a un municipio.

Con la capa de núcleos y las conexiones de cada año representadas en el visor de QGis, se eliminarán todas aquellas conexiones que estén sobre los núcleos o a menos de 100 metros de su límite. Se decide ampliar la zona de filtrado de los núcleos puesto que las actuaciones en las zonas limítrofes a las poblaciones presentan los mismos problemas que las que se realizan en la propia población. La distancia de 100 metros será el resultado de iterar con el código de forma que se alcance un acuerdo entre puntos de posibles obras lineales perdidos y puntos no útiles eliminados.

El proceso a seguir para conseguir el filtrado de estas conexiones, influidas por los núcleos urbanos, es el siguiente:

- Generar un área de influencia (*buffer*) de 100 metros a la capa de núcleos urbanos, y representarla en el mismo espacio e trabajo que las conexiones.

Figura 3-2. Búffer de núcleos urbanos

- Seleccionar por localización todos los puntos que no estén contenidos, ni intersecten el búffer previamente creado (atributo “inconexo” en QGis). En la Figura 3-3 se muestran en amarillo.

Figura 3-3. Puntos "inconexos" con búffer de núcleos urbanos

- Extraer en una nueva capa los puntos que cumplan la condición anterior.

Durante la explicación de este apartado se muestran pantallas de los distintos geoalgorítmos de QGis. Cabe destacar que se hace con intención explicativa pues todo este proceso de eliminación de puntos se ha realizado mediante *scripts* escritos en código Python, en los que se utilizan estas herramientas de geoproceso del programa. Como el resto de códigos, estarán incluidos en los anexos finales.

Concretamente para estas operaciones lo más eficiente es utilizar el creador de modelos de QGis, ya que de una forma rápida y sencilla podemos interconectar distintos geoalgoritmos de QGis y exportar el modelo a un script de Python.

3.1.2 Creador de Modelos

Para utilizar el creador de modelos debemos buscarlo en la caja de herramientas de procesos de QGis.

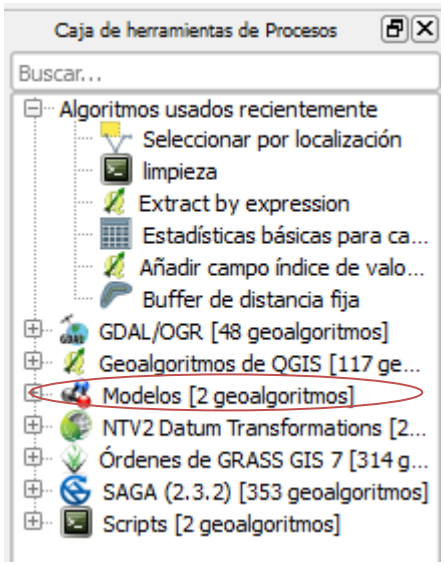


Figura 3-4. Creador de modelos

Una vez dentro del creador de modelos, seleccionamos el tipo de dato que va a recibir el modelo. Si el tipo fuera desconocido podría no concretarse, pero siempre será conveniente hacerlo.

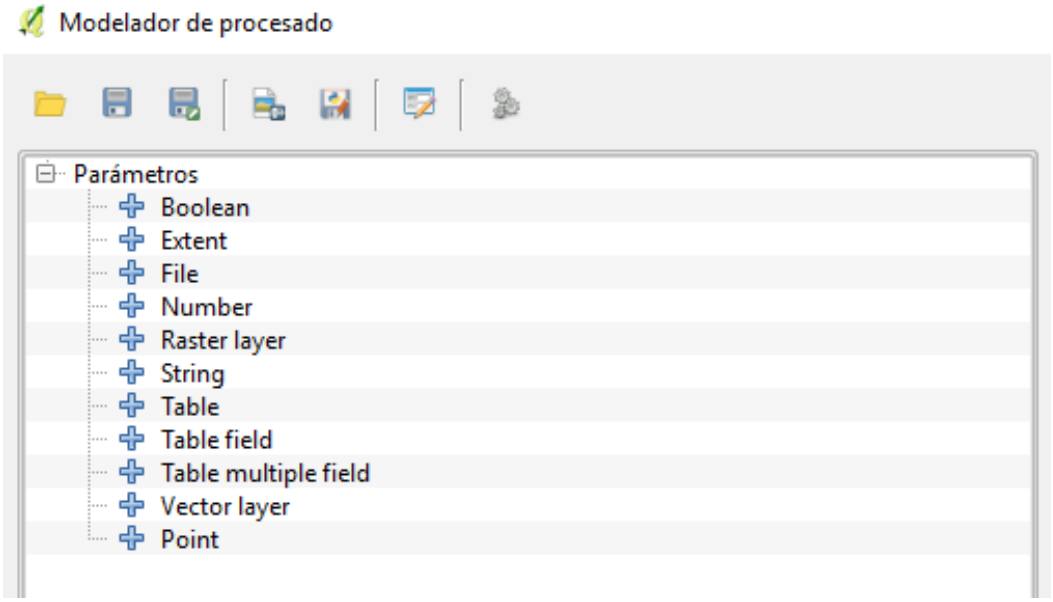


Figura 3-5. Tipos de datos de un modelo

Después se van encadenando los geoalgoritmos que vayamos a utilizar, introduciendo en cada uno de ellos las opciones elegidas y las entradas y salidas de cada uno de ellos.

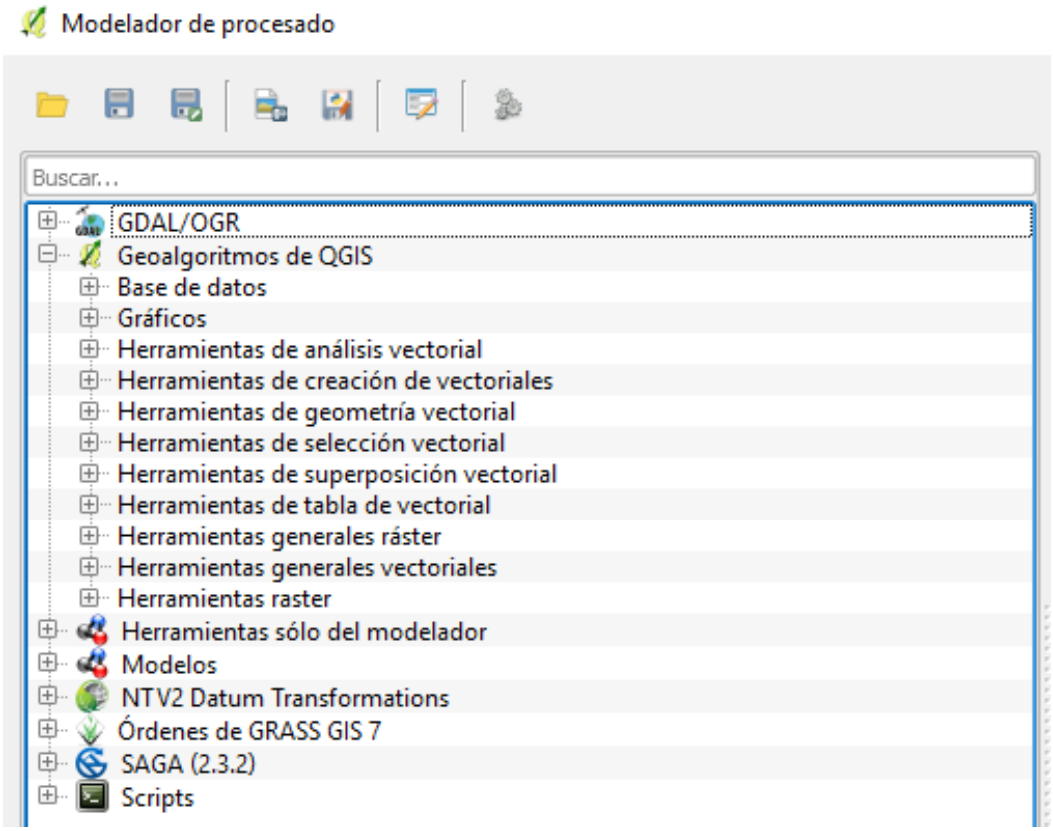


Figura 3-6. Algoritmos de QGis para modelos

Una vez estén todos los geoalgoritmos siguiendo el proceso elegido se puede exportar el modelo a Python pulsando el botón de la figura. El código generado finalmente se podrá ver en los anexos de código.

Es muy importante saber que en este caso es posible y conveniente el uso del creador de modelos por el ahorro de tiempo y porque todos los algoritmos y/o módulos utilizados están ya implementados en QGis.

A continuación, se muestra el diagrama de flujo que representa el filtrado de los datos.

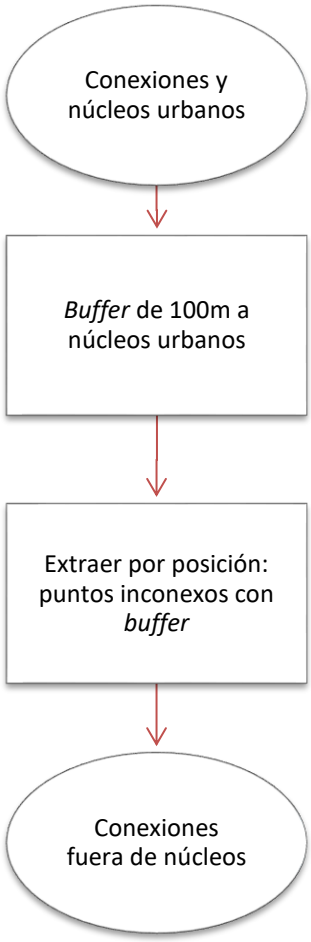


Figura 3-7. Diagrama zona urbana

3.1.3 Filtrado de Puntos Aislados

Aparte de las conexiones pertenecientes a zonas de influencia de núcleos urbanos, habrá otros puntos que no serán de interés. Estos serán los puntos aislados, es decir aquello que no tenga otro punto a su alrededor. Estas conexiones pueden deberse a diversos factores, los más habituales serán los siguientes:

- Errores en la comunicación con el servicio que provoquen una variación en las coordenadas del usuario.
- Utilización del servicio por parte de usuarios con intenciones instructivas o pruebas del servicio. Es decir, serán puntos que no formen parte de un proyecto mayor.
- Puntos de trabajos desarrollados en un día.

Para identificar aquellos puntos que consideraremos aislados, habrá que elegir una distancia respecto a la cual un punto será tomado como "punto aislado". En este caso, y tras un estudio previo exhaustivo de todos los años, se establece que una distancia de 5 km es la más adecuada.

El procedimiento será análogo al llevado a cabo para filtrar las conexiones pertenecientes a núcleos urbanos. Por ello, los pasos a realizar para eliminar los puntos aislados serán los siguientes:

- Generar un área de influencia a una distancia de 5km para cada uno de los puntos de cada año.

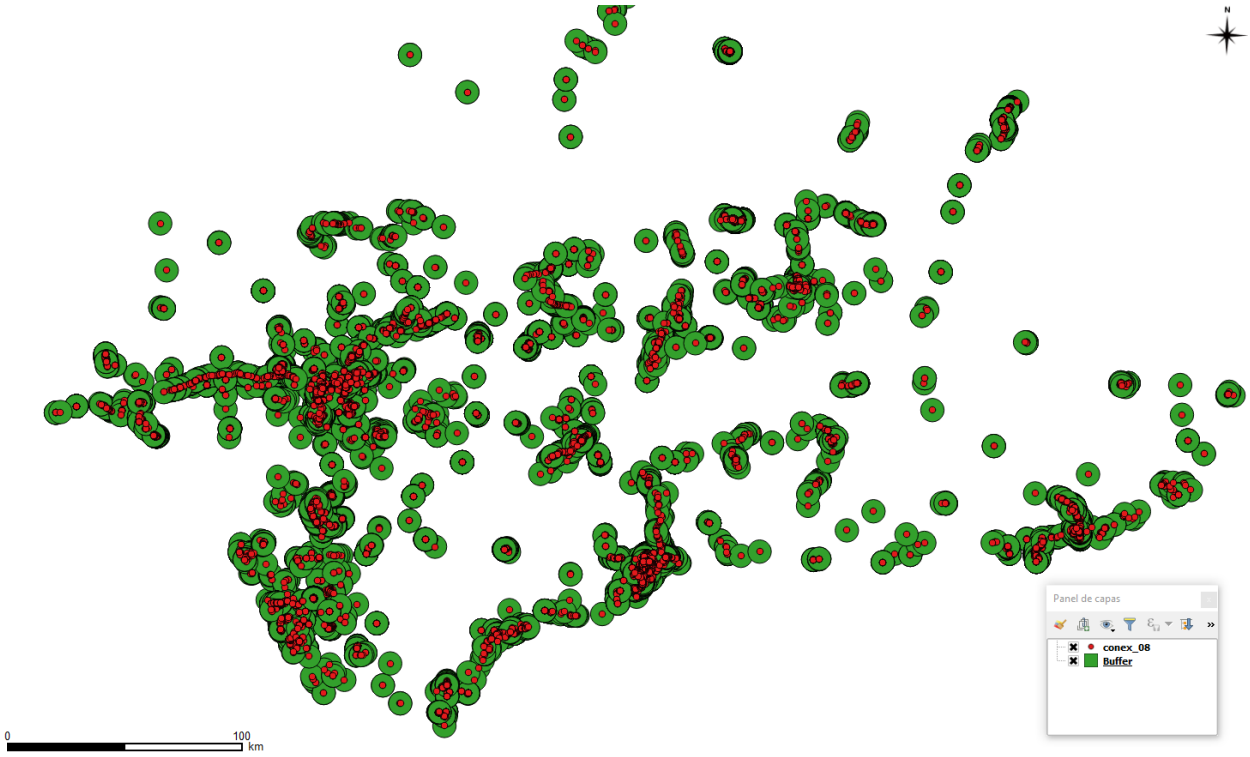


Figura 3-8. Búffer de puntos aislados

- Contar los puntos bajo los polígonos generados por el búffer, creando una nueva capa en ellos llamada "NUMPOINTS".
- Seleccionar y extraer todos aquellos polígonos del búffer cuyo atributo "*NUMPOINTS*" sea igual a 1, es decir, todos aquellos que únicamente tengan un punto bajo ellos. En la Figura 3-9 se muestran los polígonos seleccionados.

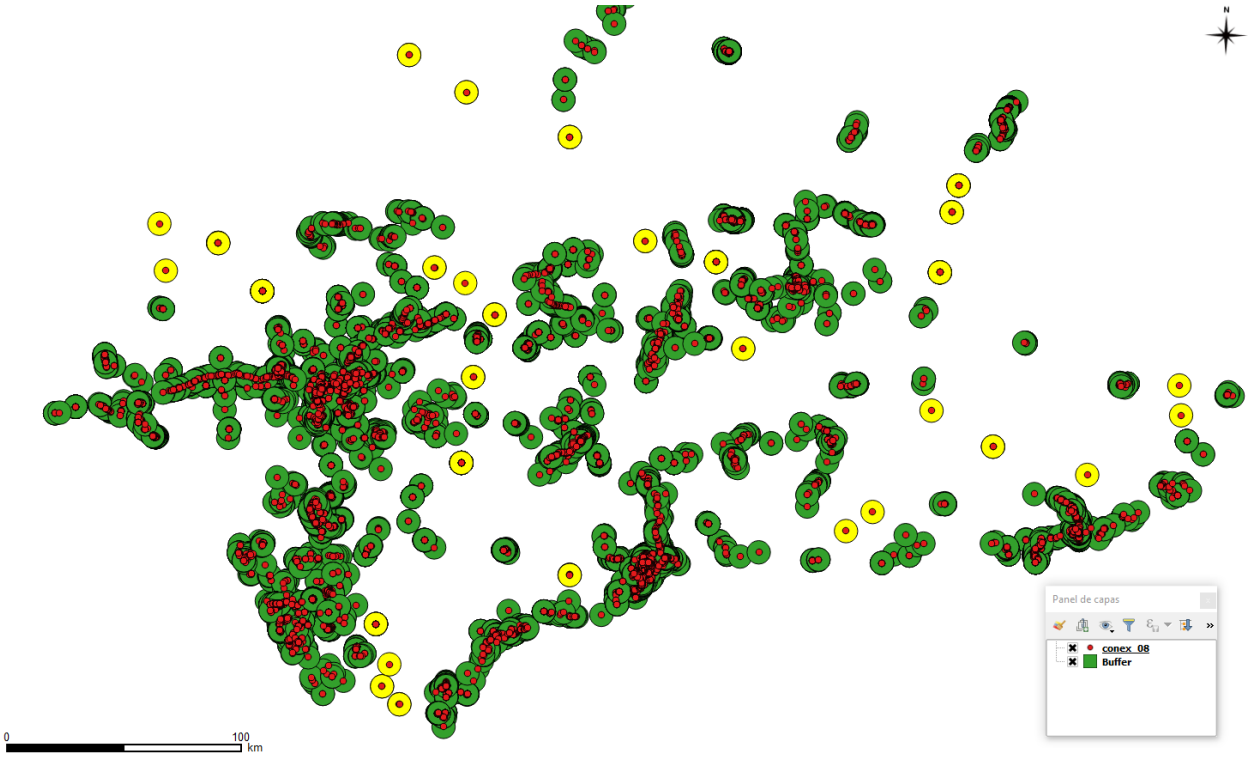


Figura 3-9. Puntos aislados

- Extraer aquellos puntos que no toque ni pertenezcan (inconexo) a alguno de los polígonos extraídos en el punto anterior.

Al igual que en el filtrado de puntos, todo este proceso se realizará mediante el creador de modelos. De hecho, se unificarán ambas operaciones en un mismo modelo, que se muestra a continuación.

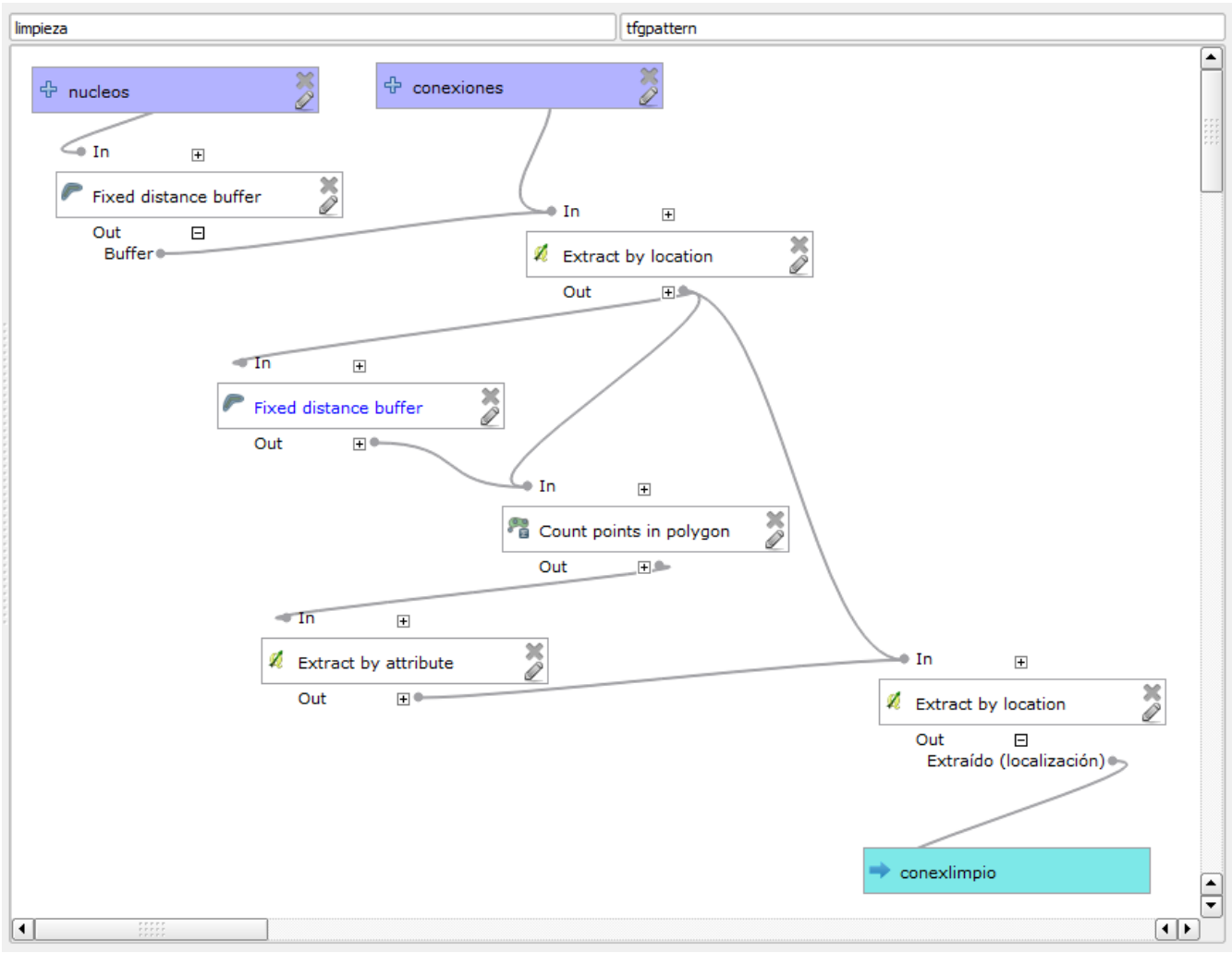


Figura 3-10. Modelo “limpieza”

Será este modelo el que se exporte a código Python, y el que se mostrará en los anexos.

3.2 Identificación de Líneas por Usuario

En este apartado se abordará la metodología seguida para identificar y seleccionar todos aquellos puntos pertenecientes a conexiones que pudieran integrar parte de una obra lineal.

En primer lugar, habrá que tener en cuenta que ahora no se podrá utilizar el creador de modelos, por lo tanto, habrá que tener en cuenta las particularidades que implica programar directamente con Python en QGis. Algunas de estas particularidades serán las siguientes:

- Para acceder a una capa y sus atributos, debe estar cargada previamente en el programa, y se asociará a una variable tipo capa (layer).
- La forma de acceder a los atributos de los distintos puntos será mediante una variable tipo iterador, es decir, para acceder a las características de un punto concreto, habrá que recorrer toda la capa.
- Si se requiere algún geoalgoritmo de QGis, habrá que importar previamente el módulo de procesado. Es decir, habrá que iniciar el código siempre con la línea: *import processing*.

- Si se obtiene alguna capa como resultado de la aplicación de algún algoritmo y se quiere guardar en una ubicación concreta, habrá que indicar la ruta elegida y, posteriormente, requerir por código el archivo presente en esa ruta concreta. Con esto se quiere decir que no se añaden por defecto al mapa activo.

Teniendo en cuenta estas particularidades, se procede a la identificación de obras lineales. Para ello se seguirá como esquema de funcionamiento el diagrama de flujo representado en la Figura 3-11. En este diagrama podemos distinguir partes claramente diferenciadas.

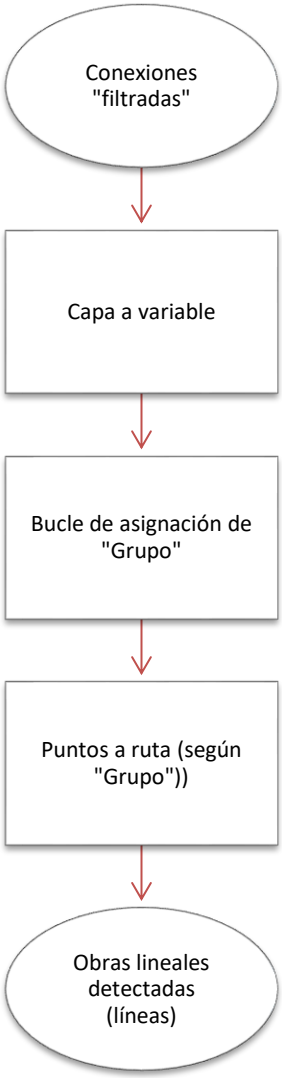


Figura 3-11. Diagrama algoritmo de identificación de obras lineales

3.2.1 Selección de capa e iniciación de contadores

La primera parte del código estará orientada a la selección de la capa de conexiones sobre la que actuar y a proporcionar un valor inicial a los contadores y variables presentes en los bucles.

La selección de la capa de conexiones se hará mediante "QgsMapLayerRegistry", accediendo al panel de capas y seleccionando por nombre en este caso. Para cada año se introducirá el nombre de la conexión pertinente (únicamente cambiará el año). La capa seleccionada se debe introducir en una variable tipo capa (layer). La asignación se variables se realizará como en la mayoría de lenguajes informáticos mediante el carácter "=".

Tan importante como la selección de la capa será asignar un objeto iterador para que recorra todos los puntos de la capa. En QGis esto se hace mediante la orden "getFeatures". Este iterador se utilizará en dos bucles, un primero para calcular la longitud de la capa en cuestión y un segundo para la identificación de obras lineales propiamente dicha.

Previamente al segundo bucle, se procede a la inicialización de los contadores. En nuestro caso, se tratará de los contadores "grupo" y "counter":

- "grupo": será el nuevo campo que identifique a los puntos que pertenezcan a una misma obra. Es decir, los puntos que pertenezcan a una misma actuación tendrán en este campo el mismo valor numérico.
- "counter": será un contador que tendrá una unidad menos que el propio iterador con el que recorremos el bucle. Esto será así para poder comparar, dentro de una misma capa, un punto de ella y justo el anterior.

3.2.2 Bucle con condición de control

El siguiente conjunto de ordenes estarán orientadas a recorrer las capas y relacionar de alguna forma aquellos puntos que puedan pertenecer a las mismas actuaciones.

La Figura 3-12 representará con mayor claridad esta parte del código

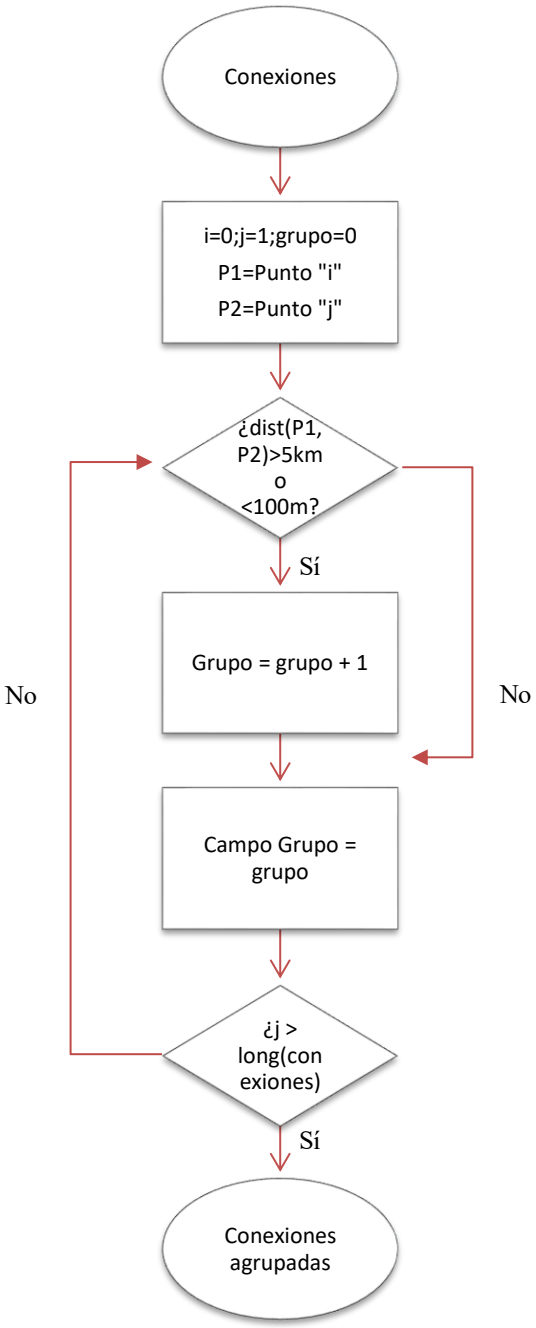


Figura 3-12. Diagrama "Grupo"

Como vemos en el diagrama, en primer lugar, se inician dos contadores basados en el mencionado anteriormente "counter". Cada uno corresponderá a los puntos que se comparan en cada iteración. Justo después de esto se hace una comprobación por medio de una condición (If) que permita salir del bucle cuando uno de los contadores sea mayor al número total de puntos.

Posteriormente se toman los valores de X e Y de los dos puntos presentes en la iteración. Estas coordenadas se introducen en variables mediante las cuales se calcula la distancia entre un punto y otro:

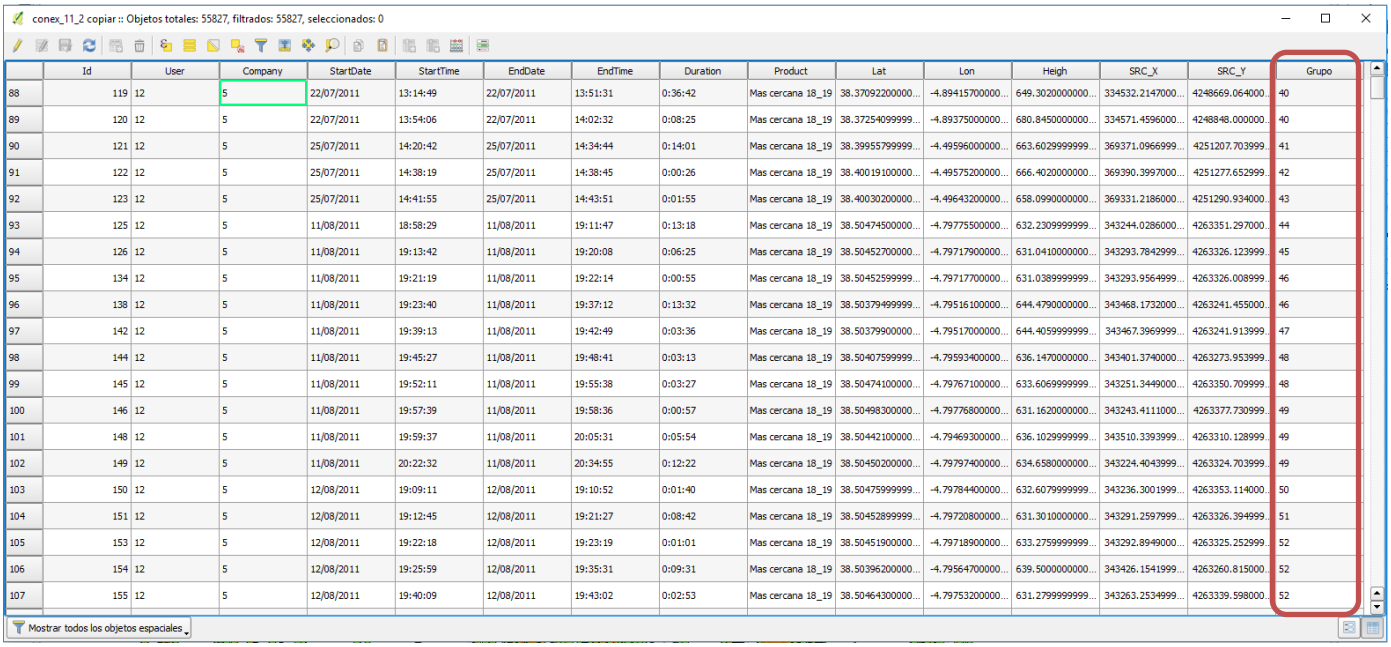
$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

siendo $(x_1, y_1), (x_2, y_2)$las coordenadas de dos puntos consecutivo. Con esta distancia, se pasa a una orden de condición para verificar si estos puntos pueden pertenecer o no a la misma actuación. Para que ambos puntos se consideren no relacionados entre ellos, deberían cumplir al menos una de las siguientes condiciones:

- Los usuarios de los puntos difieren uno del otro. Esto se daría entre puntos tomados por diferentes personas e incluso por distintas empresas, por lo que lo normal sería que no pertenecieran a la misma obra.
- La distancia entre los puntos es superior a 5 km. Como se decidió para elegir la distancia a partir de la cual un punto se consideraría aislado, parece lógico pensar que puntos alejados esa distancia pertenecerán a actuaciones distintas.
- La distancia entre los puntos es inferior a 100 m. Analizando las conexiones, se puede llegar a la conclusión de que puntos más cercanos de 100 m pueden ser consecuencia de solapación entre distintas actuaciones o estudios en una pequeña zona que no tendrán nada que ver con obras lineales.

Siempre que se cumpla alguna de estas condiciones, el contador de "grupo" aumentará en una unidad, de forma que los siguientes puntos no queden relacionados con los anteriores.

De esta forma, los puntos quedarían relacionados por el nuevo campo "Grupo".



	Id	User	Company	StartDate	StartTime	EndDate	EndTime	Duration	Product	Lat	Lon	Heigh	SRC_X	SRC_Y	Grupo
88	119	12	5	22/07/2011	13:14:49	22/07/2011	13:51:31	0:36:42	Mas cercana 18_19	38.37092200000...	-4.89415700000...	649.3020000000...	334532.2147000...	4248669.064000...	40
89	120	12	5	22/07/2011	13:54:06	22/07/2011	14:02:32	0:08:25	Mas cercana 18_19	38.37254099999...	-4.89375000000...	680.8450000000...	334571.4596000...	4248848.000000...	40
90	121	12	5	25/07/2011	14:20:42	25/07/2011	14:34:44	0:14:01	Mas cercana 18_19	38.39955799999...	-4.49596000000...	663.6029999999...	369371.0966999...	4251207.703999...	41
91	122	12	5	25/07/2011	14:38:19	25/07/2011	14:38:45	0:00:26	Mas cercana 18_19	38.40019100000...	-4.49575200000...	666.4020000000...	369390.3997000...	4251277.652999...	42
92	123	12	5	25/07/2011	14:41:55	25/07/2011	14:43:51	0:01:55	Mas cercana 18_19	38.40030200000...	-4.49643200000...	658.0990000000...	369331.2186000...	4251290.934000...	43
93	125	12	5	11/08/2011	18:58:29	11/08/2011	19:11:47	0:13:18	Mas cercana 18_19	38.50474500000...	-4.79775500000...	632.2309999999...	343244.0286000...	4263351.297000...	44
94	126	12	5	11/08/2011	19:13:42	11/08/2011	19:20:08	0:06:25	Mas cercana 18_19	38.50452700000...	-4.79717900000...	631.0410000000...	343293.7842999...	4263326.123999...	45
95	134	12	5	11/08/2011	19:21:19	11/08/2011	19:22:14	0:00:55	Mas cercana 18_19	38.50452599999...	-4.79717700000...	631.0389999999...	343293.9564999...	4263326.008999...	46
96	138	12	5	11/08/2011	19:23:40	11/08/2011	19:37:12	0:13:32	Mas cercana 18_19	38.50379499999...	-4.79516100000...	644.4790000000...	343468.1732000...	4263241.455000...	46
97	142	12	5	11/08/2011	19:39:13	11/08/2011	19:42:49	0:03:36	Mas cercana 18_19	38.50379900000...	-4.79517000000...	644.4059999999...	343467.3969999...	4263241.913999...	47
98	144	12	5	11/08/2011	19:45:27	11/08/2011	19:48:41	0:03:13	Mas cercana 18_19	38.50407599999...	-4.79593400000...	636.1470000000...	343401.3740000...	4263273.953999...	48
99	145	12	5	11/08/2011	19:52:11	11/08/2011	19:55:38	0:03:27	Mas cercana 18_19	38.50474100000...	-4.79767100000...	633.6069999999...	343251.3449000...	4263350.709999...	48
100	146	12	5	11/08/2011	19:57:39	11/08/2011	19:58:36	0:00:57	Mas cercana 18_19	38.50498300000...	-4.79776800000...	631.1620000000...	343243.4111000...	4263377.730999...	49
101	148	12	5	11/08/2011	19:59:37	11/08/2011	20:05:31	0:05:54	Mas cercana 18_19	38.50442100000...	-4.79469300000...	636.1029999999...	343510.3393999...	4263310.128999...	49
102	149	12	5	11/08/2011	20:22:32	11/08/2011	20:34:55	0:12:22	Mas cercana 18_19	38.50450200000...	-4.79797400000...	634.6580000000...	343224.4043999...	4263324.703999...	49
103	150	12	5	12/08/2011	19:09:11	12/08/2011	19:10:52	0:01:40	Mas cercana 18_19	38.50475999999...	-4.79784400000...	632.6079999999...	343236.3001999...	4263353.114000...	50
104	151	12	5	12/08/2011	19:12:45	12/08/2011	19:21:27	0:08:42	Mas cercana 18_19	38.50452899999...	-4.79720800000...	631.3010000000...	343291.2597999...	4263326.394999...	51
105	153	12	5	12/08/2011	19:22:18	12/08/2011	19:23:19	0:01:01	Mas cercana 18_19	38.50451900000...	-4.79718900000...	633.2759999999...	343292.8949000...	4263325.252999...	52
106	154	12	5	12/08/2011	19:25:59	12/08/2011	19:35:31	0:09:31	Mas cercana 18_19	38.50396200000...	-4.79564700000...	639.5000000000...	343426.1541999...	4263260.815000...	52
107	155	12	5	12/08/2011	19:40:09	12/08/2011	19:43:02	0:02:53	Mas cercana 18_19	38.50464300000...	-4.79753200000...	631.2799999999...	343263.2534999...	4263339.598000...	52

Figura 3-13. Campo "grupo"

3.2.3 Puntos a ruta

Por último, con los puntos clasificados por su nuevo atributo "Grupo", se trazarán las líneas entre todos aquellos puntos que formen parte de la misa obra lineal. Para esta tarea, lo más eficiente es utilizar el geoalgoritmo de QGis "Puntos a ruta" (pointstopath).

Este algoritmo traza líneas a partir de capas de puntos según un determinado atributo. En nuestro caso, se elegirá el atributo "Grupo". Además, el geoalgoritmo requiere algún otro atributo mediante el cual establecer el orden

de trazado de las líneas. En nuestro caso se utilizará el campo "Id" para dicha ordenación, puesto que este mantendrá en orden temporal (recordemos que se estableció previamente).

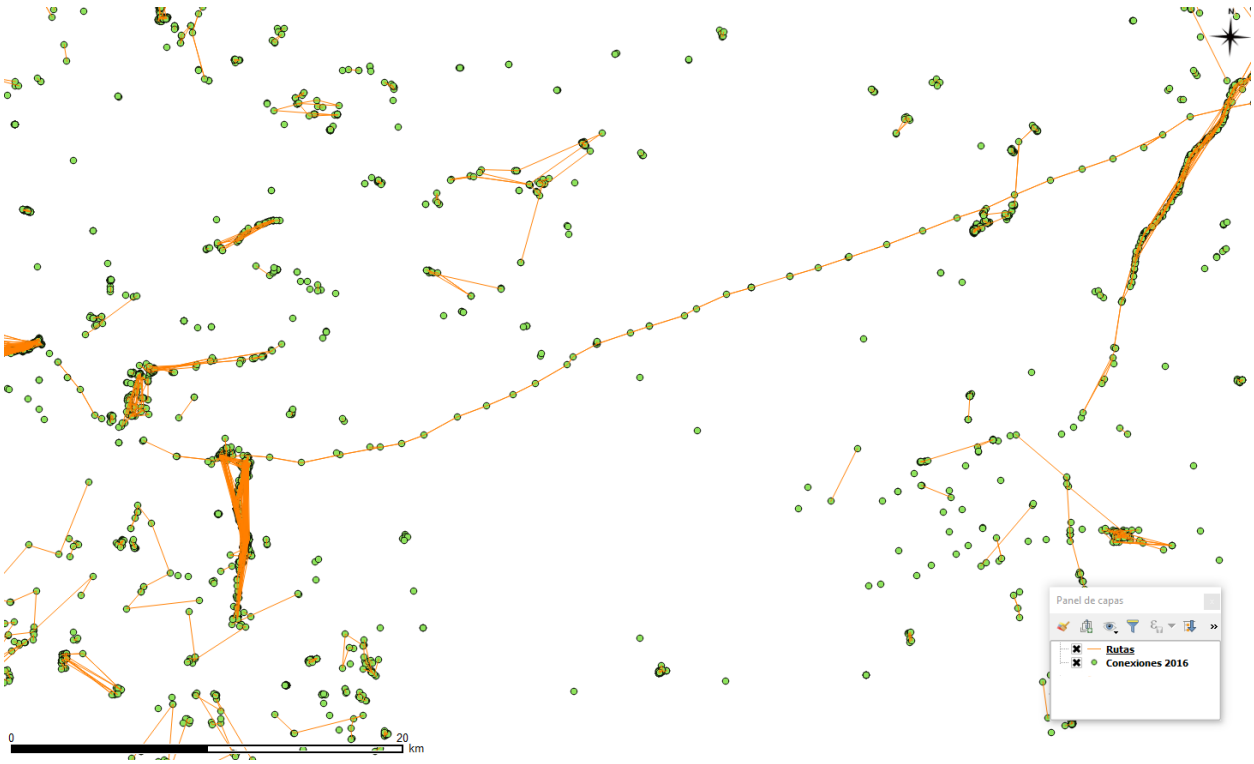


Figura 3-14. Puntos a ruta

Una vez utilizado el algoritmo, habremos generado una nueva capa con las posibles actuaciones identificadas, en la que cada línea tendrá el número correspondiente al campo "Grupo" de cuyos puntos integran la propia línea.

3.3 Filtrado por Sinuosidad

Una vez aplicado el algoritmo de identificación de obras lineales, nos encontramos con muchos casos de líneas que claramente no pertenecen a obras lineales, o incluso casos que no parecen tener ninguna relación lógica. Para limpiar el mapa de todas estas líneas, o al menos conseguir eliminar las máximas posibles, se idea un nuevo algoritmo capaz de extraer líneas que tengan características más propias de las obras lineales.

La idea en este caso se basa en crear un nuevo campo "Sinuosidad", que mida de alguna forma la diferencia entre cada línea y una línea completamente recta. La fórmula que aplicaremos en este nuevo campo será:

$$\frac{Longitud\ real\ de\ la\ curva}{Distancia\ entre\ origen\ y\ fin} = \frac{L}{\sqrt{\left(x_{fin} - x_{ini}\right)^2 + \left(y_{fin} - y_{ini}\right)^2}}$$

Según esta fórmula, cuanto más se alejen las líneas del valor 1, más reviradas serán, y por tanto menos opciones tendrán de ser una obra lineal. De acuerdo con este criterio, se eliminarán todas aquellas líneas que tengan un valor de sinuosidad inferior a algún valor representativo.

Para hallar el valor de referencia se hacen diferentes pruebas con distintos valores. Finalmente se concreta que 1.1 será un valor en el que se alcanzará un acuerdo entre líneas erróneas eliminadas y líneas posibles pérdidas.

Además de todo esto, también se eliminarán todas aquellas líneas que estén compuestas por dos puntos o uno (habiendo degenerado en otro punto). La razón de esta supresión de líneas atiende a que éstas estarán más relacionadas con pequeños estudios que tendrán poco que ver con las obras lineales objetivo de este trabajo.

Finalmente, tras aplicar este nuevo algoritmo se obtendrá algo parecido a lo representado en la Figura 3-15:

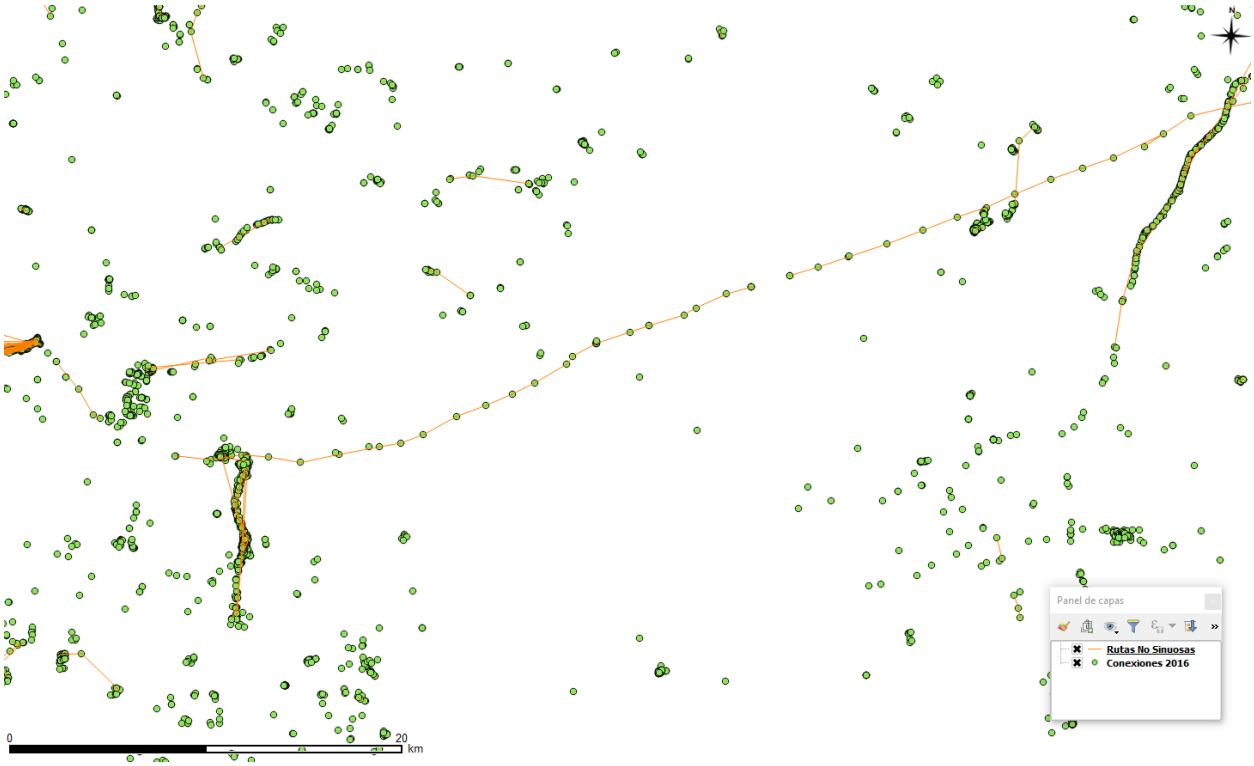


Figura 3-15. Uso de sinuosidad

4 RESULTADOS OBTENIDOS

En este apartado se procederá al análisis de los resultados obtenidos. En un primer momento las obras lineales detectadas por el algoritmo se compararán con las diferentes capas del DERA en las que consten obras y actuaciones realmente efectuadas.

Tras esta comparación, se mostrarán y analizarán todas aquellas obras lineales que no pertenezcan a ninguna de esas capas, tratando de determinar de qué se trata en cada caso.

4.1 Comparación con DERA

Para realizar esta comparación, en primer lugar, habrá que determinar que capas serán de utilidad para dicha tarea. Puesto que el estudio va orientado a obras lineales se tomarán todas aquellas capas que representen este tipo de elementos. Con tal intención se tomarán las representadas en la Tabla 4-1:

Tabla 4-1. Capas DERA

Nombre	Descripción	Fecha
vc01_carretera_arco	Red de carreteras	24/06/2013
vc03_ffcc	Red de ferrocarriles	21/02/2014
ie06_elect_linea	Líneas eléctricas	27/06/2018
ie10_gas_gasoducto	Gasoductos	27/06/2018
ih03_conduccion	Sistema de conducciones	01/11/2014

Como se explicó anteriormente, estas capas pueden descargarse directamente en formato shapefile (.shp) de la página web del DERA. Una vez conseguidas las capas buscadas, podrán importarse al visor de QGis.

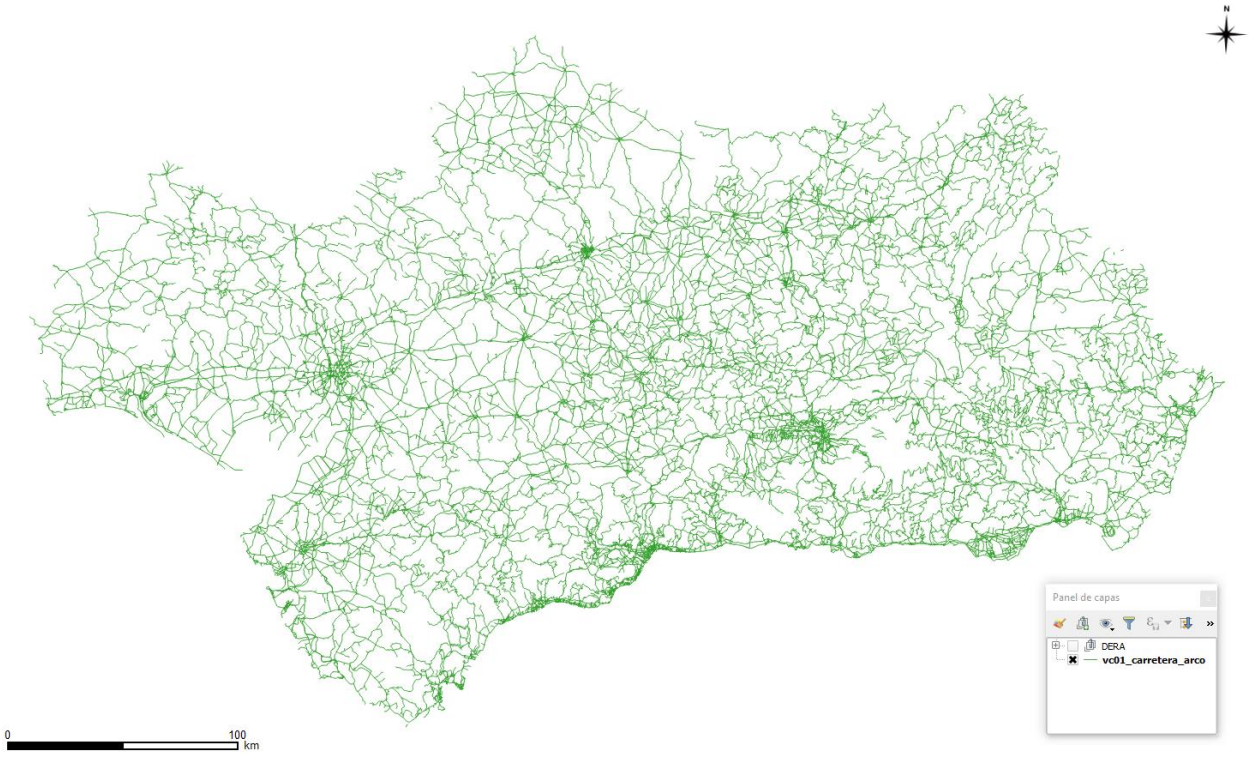


Figura 4-1. Carreteras

Figura 4-2. Ferrocarriles

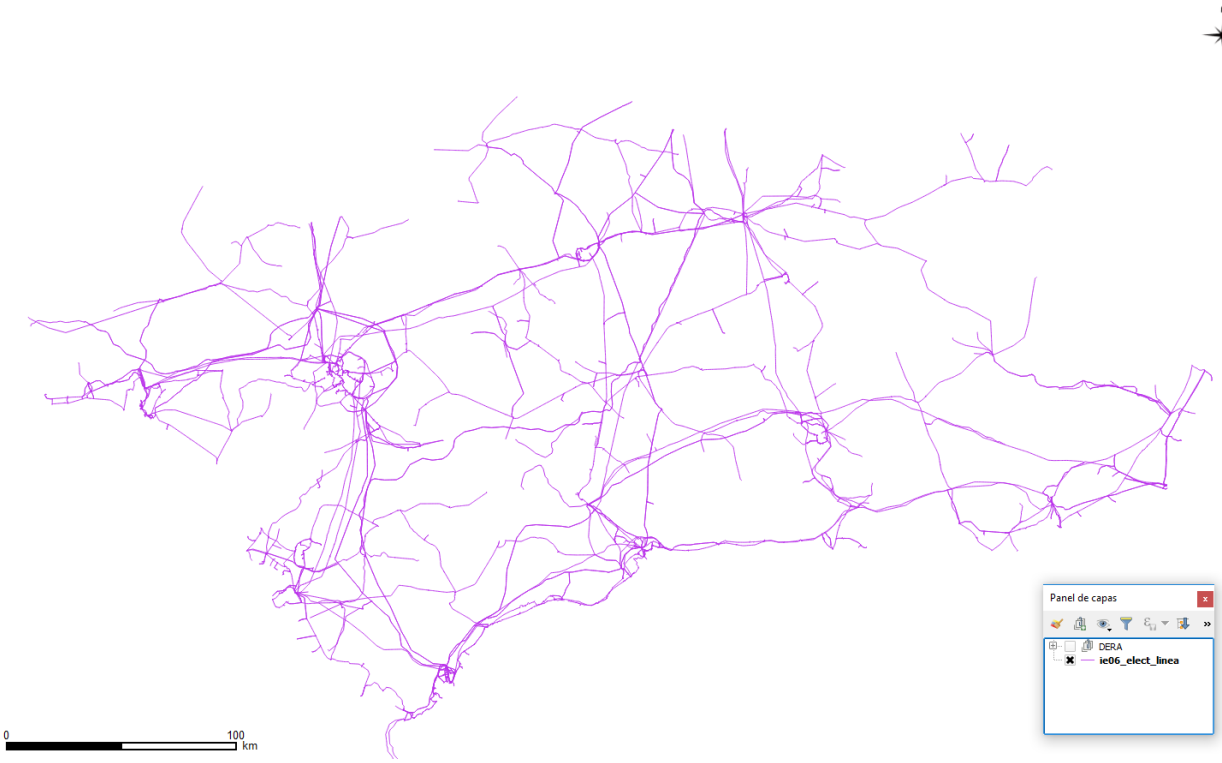


Figura 4-3. Líneas Eléctricas

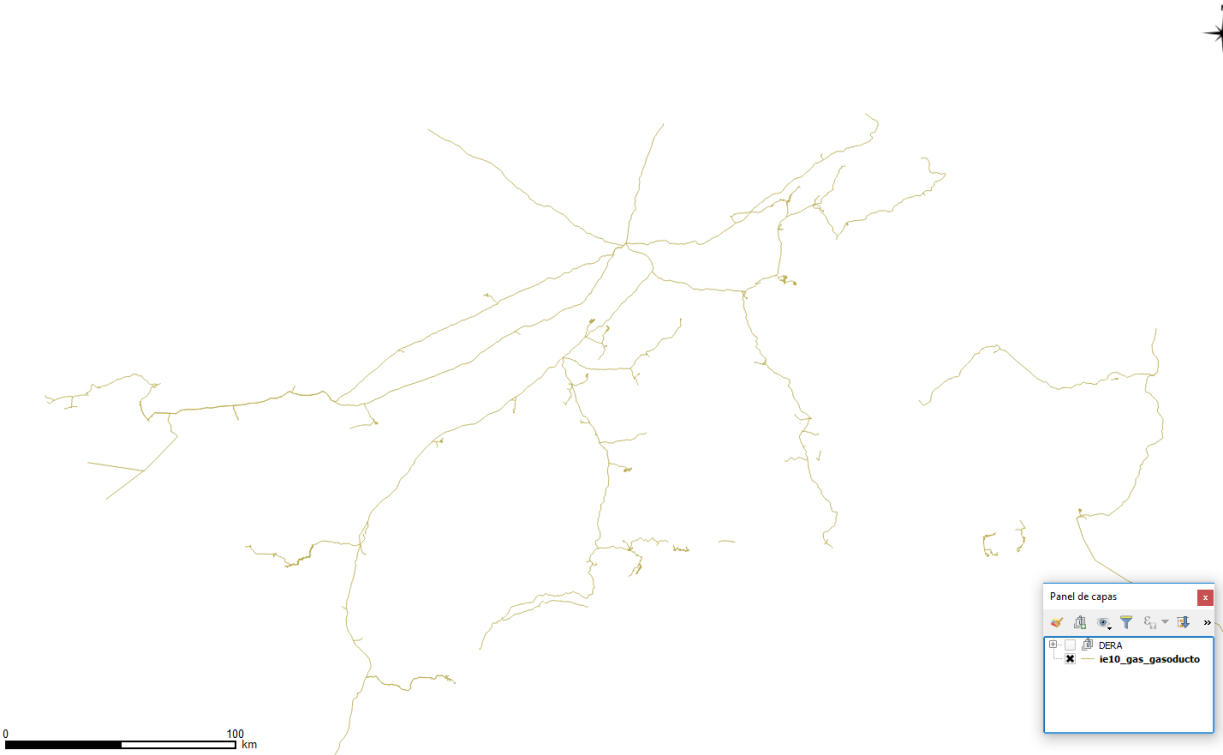


Figura 4-4. Gasoductos

Figura 4-5. Conducciones

Con las capas cargadas en el visor de QGis, se podrá utilizar un algoritmo para detectar aquellas líneas ("rutas") resultado del *script* de identificación de patrones lineales, que pertenezcan a cada una de las diferentes categorías. Las categorías quedarán delimitadas por las diferentes capas del DERA.

Para generar dicho algoritmo se volverá a recurrir al Creador de Modelos, exportándose el resultado a código Python. Ya se describió en capítulos anteriores cómo se realizaba esto. El código utilizado se mostrará en el anexo correspondiente. En este caso, el código se regirá por el siguiente diagrama de flujo:

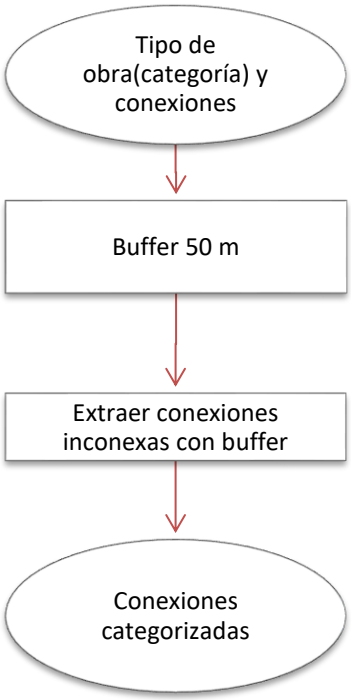


Figura 4-6. Diagrama Flujo: Categorización

Una vez ejecutado este algoritmo para cada una de las rutas correspondientes a los diferentes años y para cada una de las categorías en cada caso, se obtendrán las rutas correspondientes a las obras existentes categorizadas por tipo. Esto nos será útil tanto para diferenciar entre posibles actuaciones nuevas, como para tratar de identificar patrones temporales o espaciales entre las conexiones de las diferentes categorías. Esto último se abordará más ampliamente en el capítulo de conclusiones.

En las Figura 4-7 y 4-8 se muestra un ejemplo ilustrativo en el que se ven representadas las rutas coincidentes con una de las categorías, en este caso gasoductos.

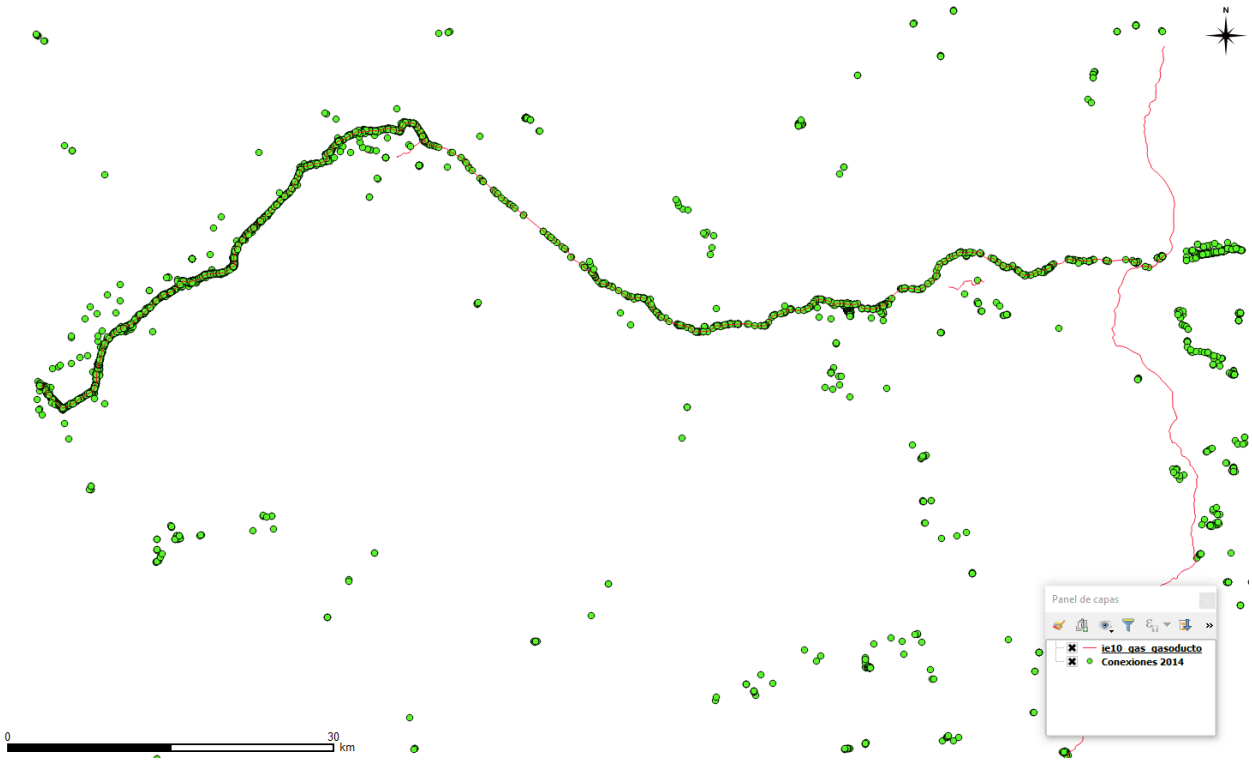


Figura 4-7. Detalle Gasoducto

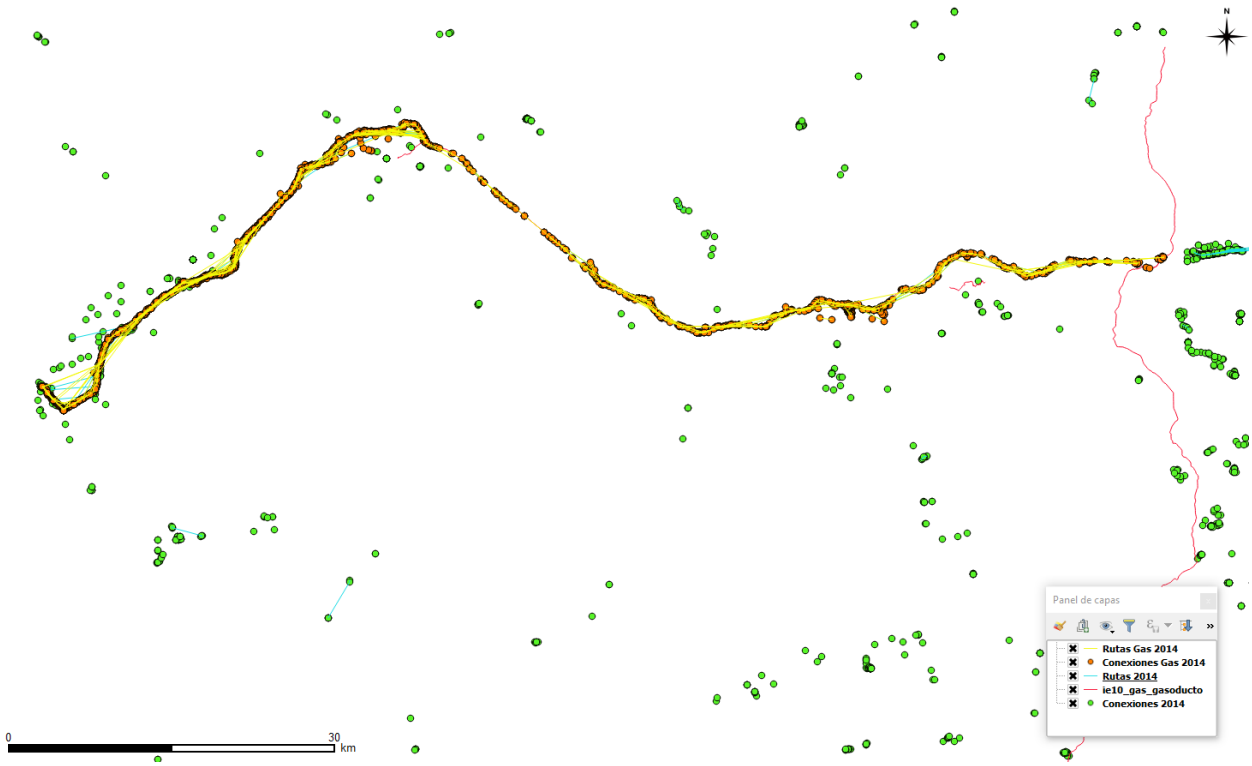


Figura 4-8. Ruta Gasoducto

4.2 Rutas no categorizadas

Después de categorizar las conexiones y sus rutas en las diferentes tipologías, se podrán extraer el resto de elementos que no pertenezcan a ninguna de estas categorías. Con esto se buscará encontrar las posibles actuaciones o variantes no reflejadas en el DERA.

Para dicha tarea se volverá a generar un proceso mediante el Creador de Modelos y exportándolo posteriormente a lenguaje Python. El proceso seguido por el programa vendrá determinado por el siguiente diagrama de flujo:

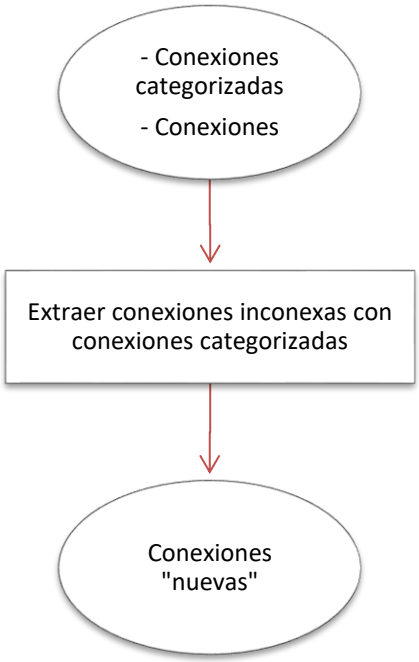


Figura 4-9. Diagrama Flujo: Obra Nueva

Tras ejecutar este algoritmo se generarán las nuevas capas de rutas (correspondientes a cada uno de los años), cuyo recorrido no se corresponda con ninguna de las categorías estudiadas. En todos los casos, seguirá habiendo muchas rutas que no serán de utilidad. Esto puede deberse a alguna de las siguientes razones:

- Actuaciones más recientes que la última actualización de la capa del DERA correspondiente.
- Vuelos de drones con sistemas de posicionamiento.
- Rutas provenientes de conexiones en parcelas particulares, sobre todo de carácter agrícola.

Por tanto, habrá que estudiar cuidadosamente cada caso para determinar si se puede tratar de una actuación no reflejada en la cartografía oficial o no. Para terminar de verificar estas variantes nuevas, se empleará la ortofotografía más reciente del PNOA (Plan Nacional de Ortofotografía Aérea).

4.3 Comparación con PNOA

Para utilizar las ortofotografías del PNOA se recurrirá al servicio WMS del mismo, mediante el cual se podrá insertar las imágenes en QGis de forma que sobre ellas se puedan disponer los elementos a revisar. Las imágenes del PNOA utilizadas, serán del vuelo ortofotográfico de 2016.

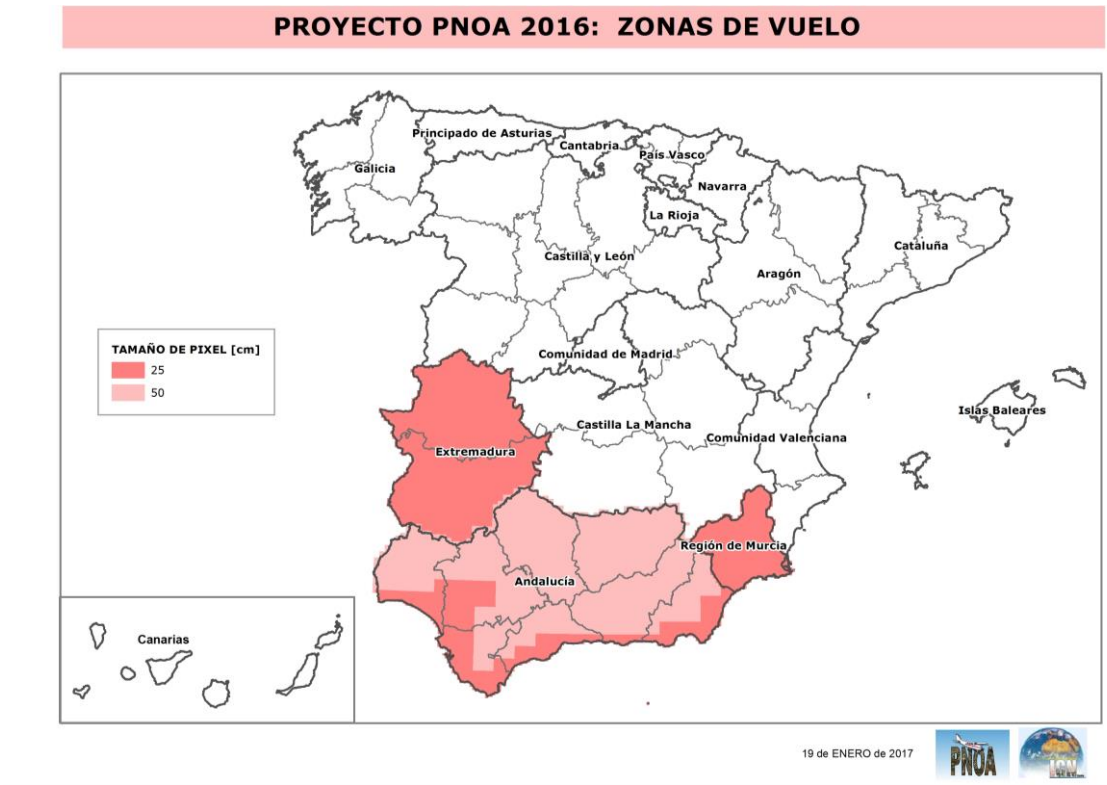


Figura 4-10. Vuelo PNOA

En los siguientes subapartados se mostrarán las rutas más destacadas agrupadas por año, discutiéndose en cada caso el posible origen de cada una.

4.3.1 Rutas "nuevas" 2008

En este año, únicamente se identificará un tramo destacable. La ubicación de la ruta se muestra en la Figura 4-11.

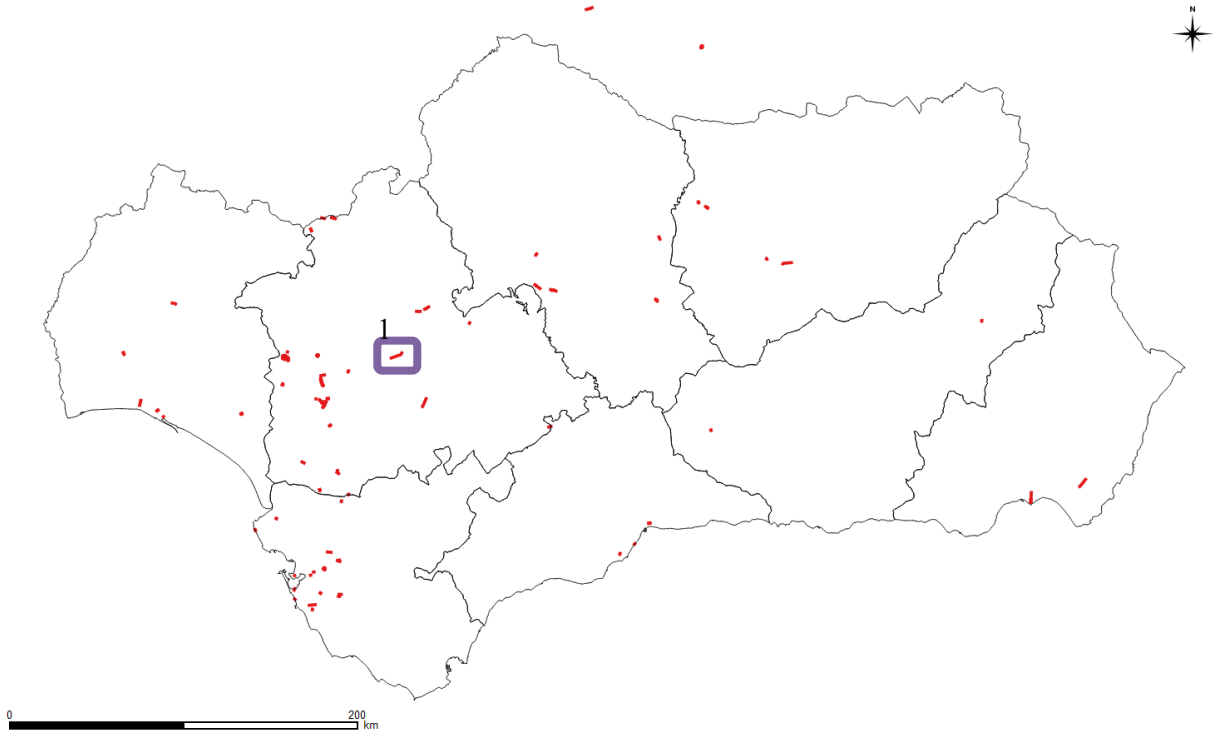


Figura 4-11. Rutas 2008

Figura 4-12. Ruta 1_2008

Tabla 4-2. Ruta 1_2008

Año	**2008**
Nº Ruta	1
Observaciones	Al no estar cerca de ninguna línea de las categorías indicadas, y debido a la proximidad de la urbanización; parece indicar que puede tratarse de parte de la red de abastecimiento /saneamiento de la misma.

4.3.2 Rutas “nuevas” 2009

De nuevo se vuelve a analizar una única ruta. La ubicamos en la Figura 4-13.

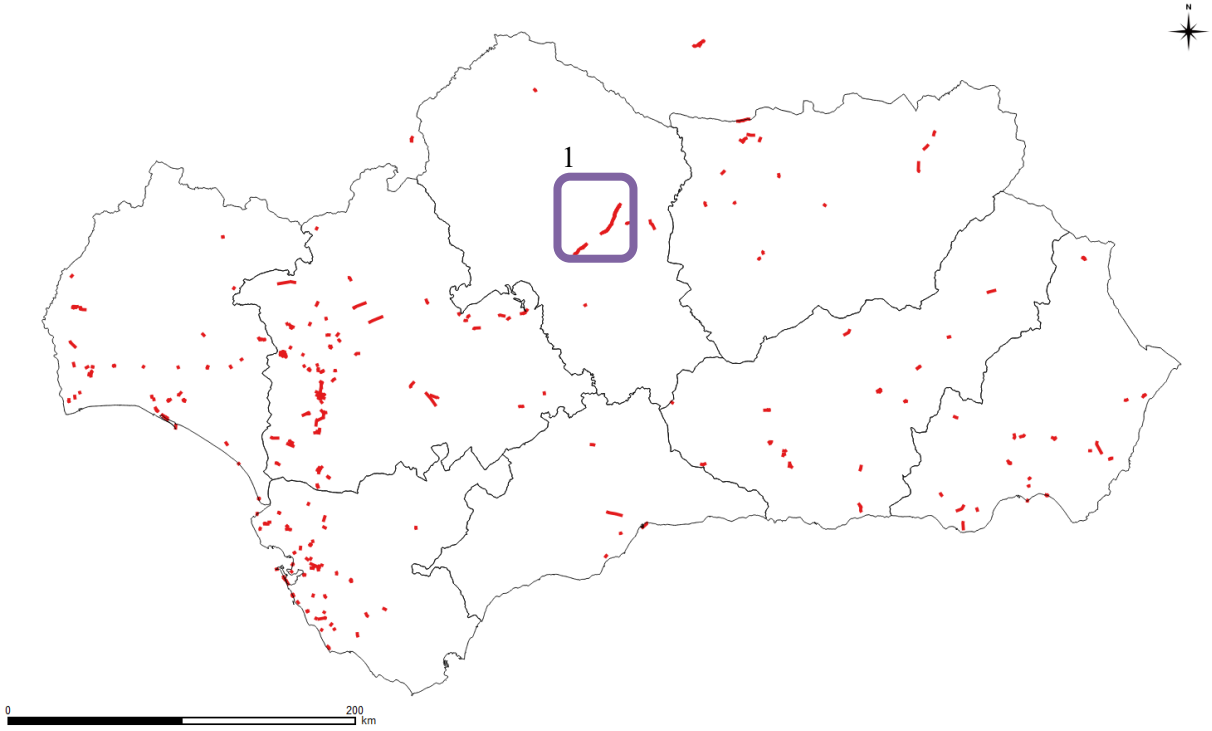


Figura 4-13. Rutas 2009

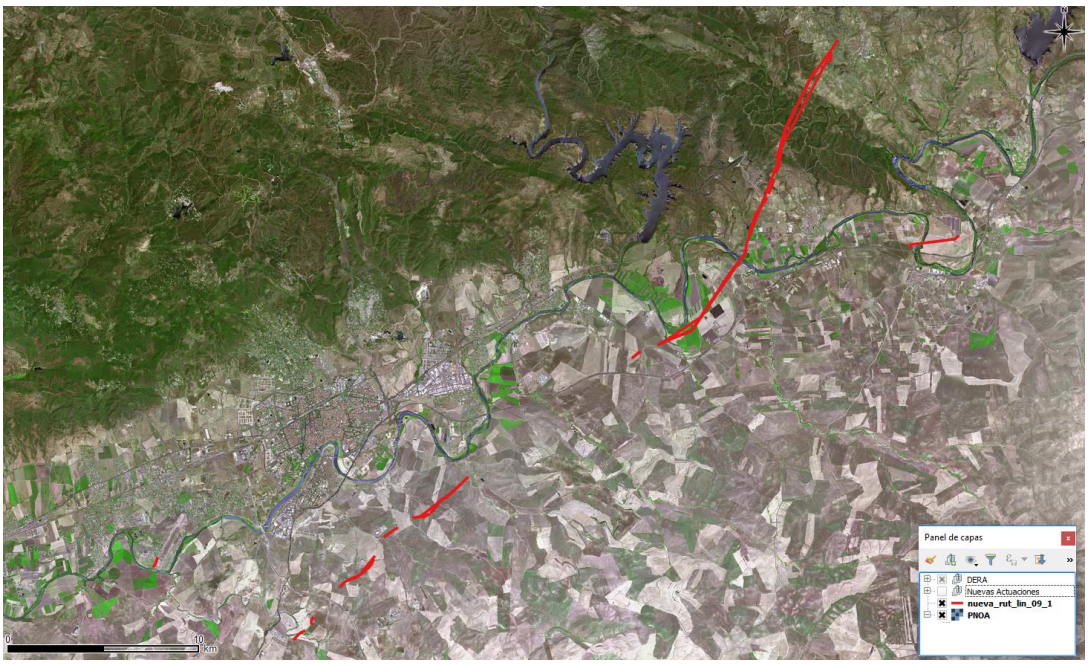


Figura 4-14. Ruta 1_2009

Tabla 4-3. Ruta 1_2009

Año	**2009**
Nº Ruta	1
Observaciones	Observando más de cerca la ortofotografía, se detecta que las líneas siguen caminos entre parcelas de la zona.

4.3.3 Rutas "nuevas" 2010

En la Figura 4-15, se observa la misma zona del año anterior.

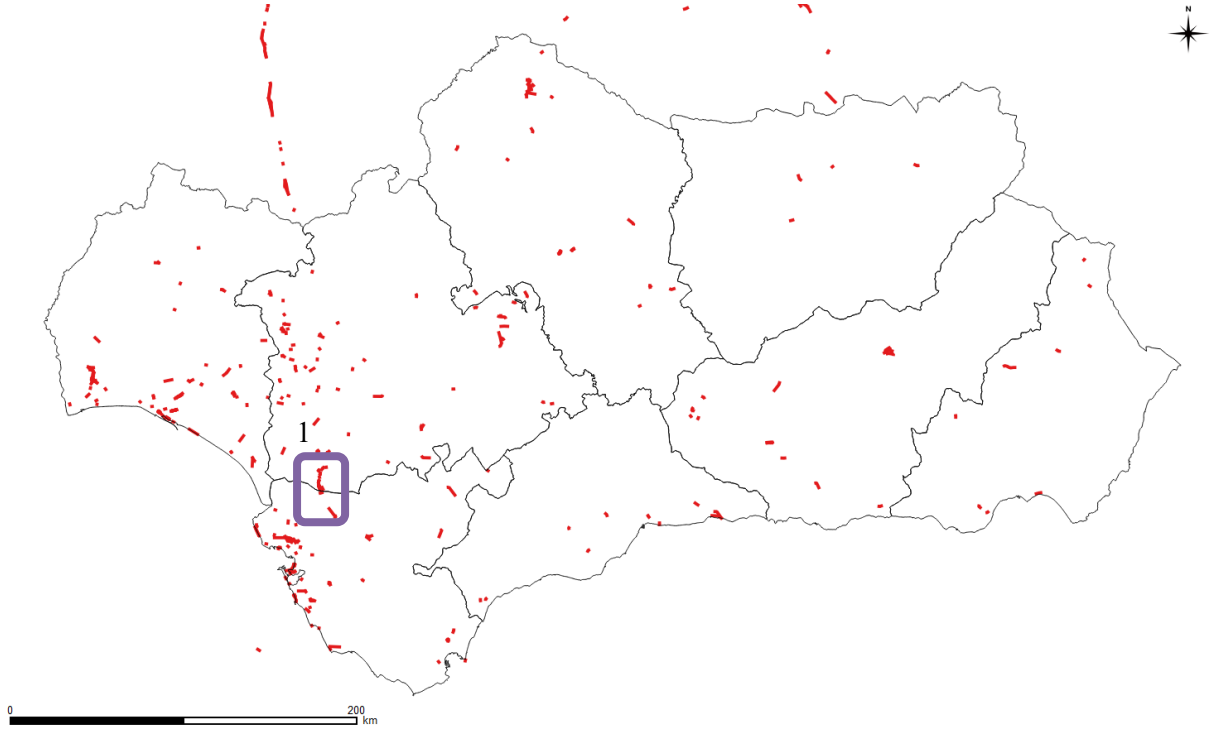


Figura 4-15. Rutas 2010

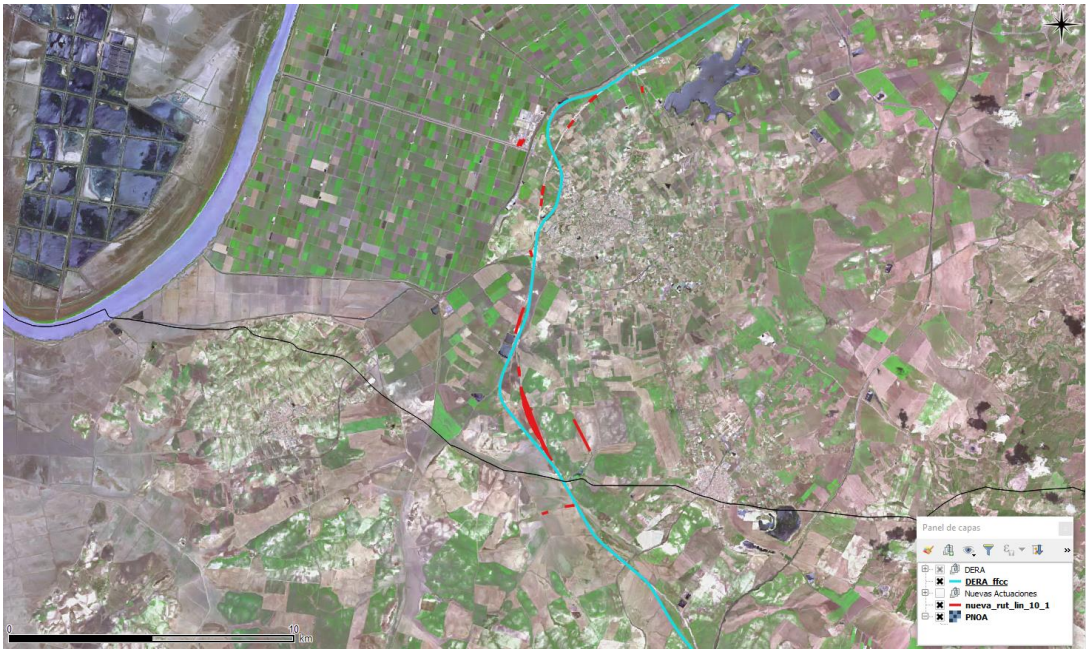


Figura 4-16.Ruta 1_2010

Tabla 4-4. Ruta 1_2010

Año	2010
Nº Ruta	1
Observaciones	Se aprecian tanto en la ortofoto, como al analizar la capa de DERA, que se trata de actuaciones en la vía Sevilla-Cádiz.

4.3.4 Rutas "nuevas" 2011

Se representan en este caso 2 rutas para analizar en la Figura 4-17.

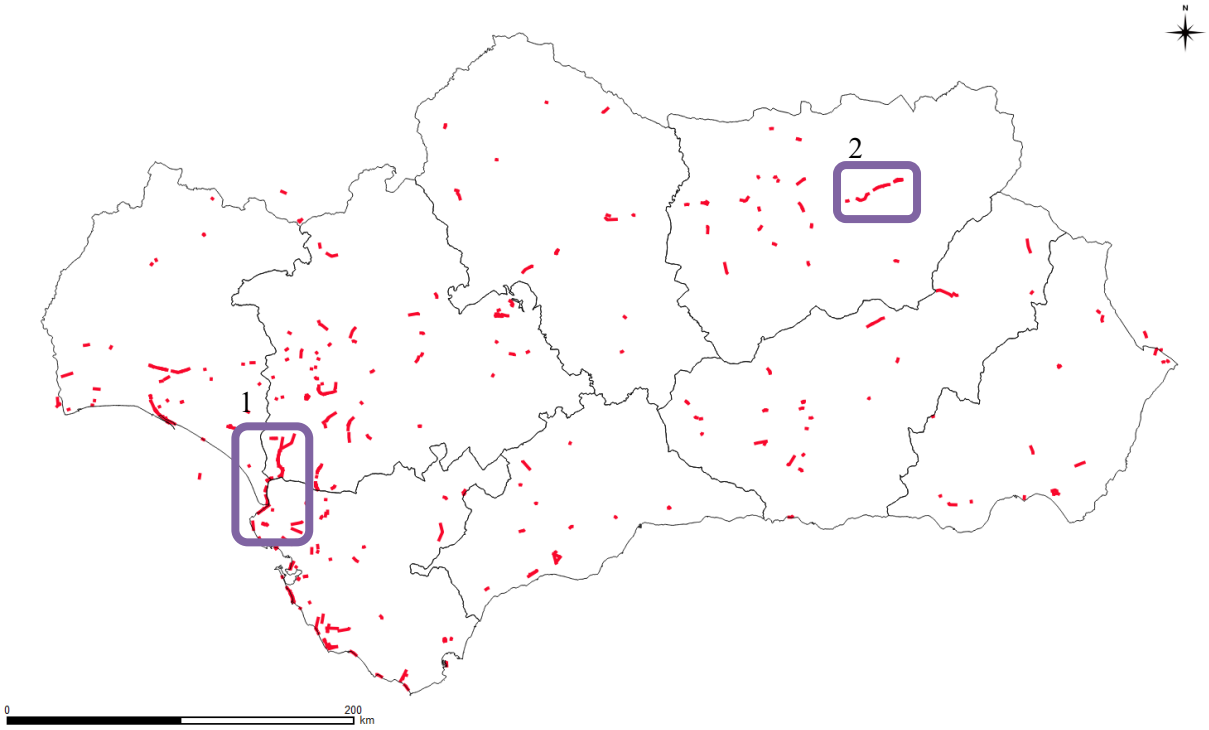


Figura 4-17. Rutas 2011

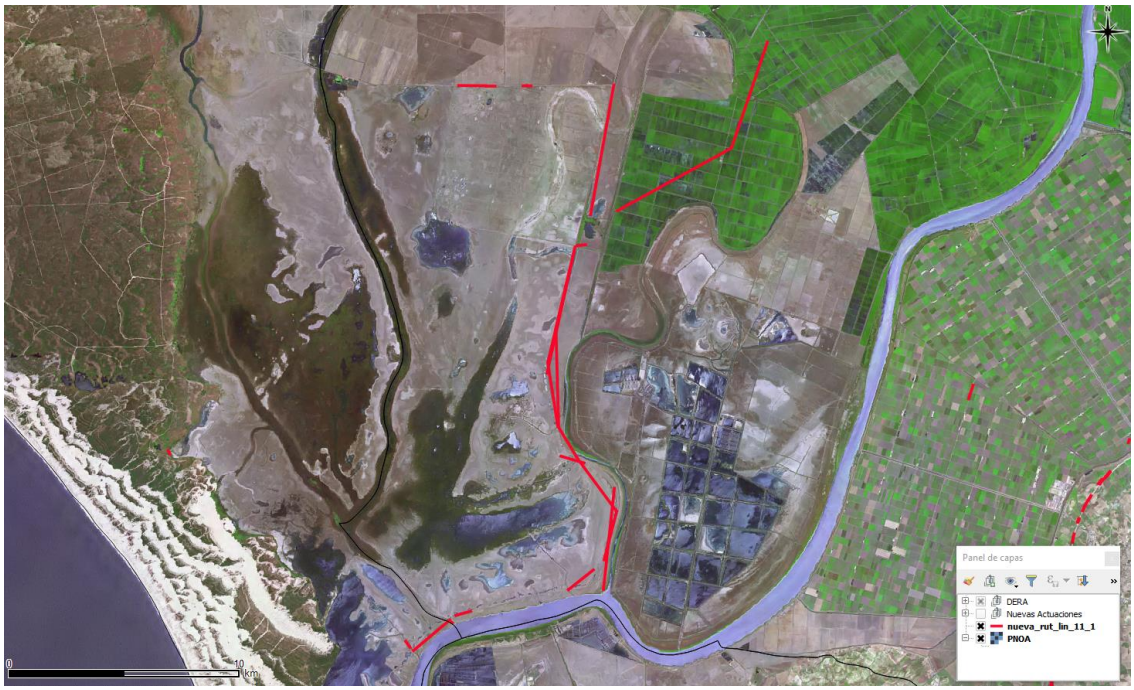


Figura 4-18. Ruta 1_2011

Tabla 4-5. Ruta 1_2011

Año	**2011**
Nº Ruta	1
Observaciones	Actuaciones en la zona del delta del Guadalquivir, concretamente creación de caminos/accesos.

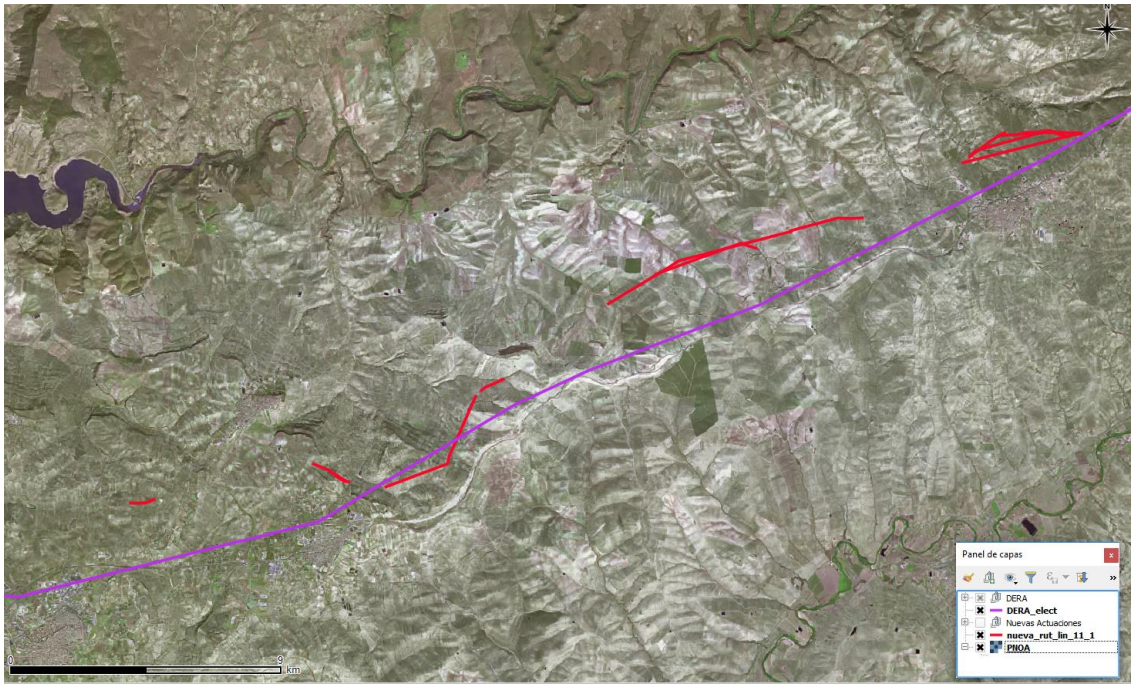


Figura 4-19. Ruta 2_2011

Tabla 4-6. Ruta 2_2011

Año	**2011**
Nº Ruta	2
Observaciones	Posibles variaciones en el trazado de una línea eléctrica

4.3.5 Rutas “nuevas” 2012

Se muestran las rutas más reseñables en la Figura 4-20.

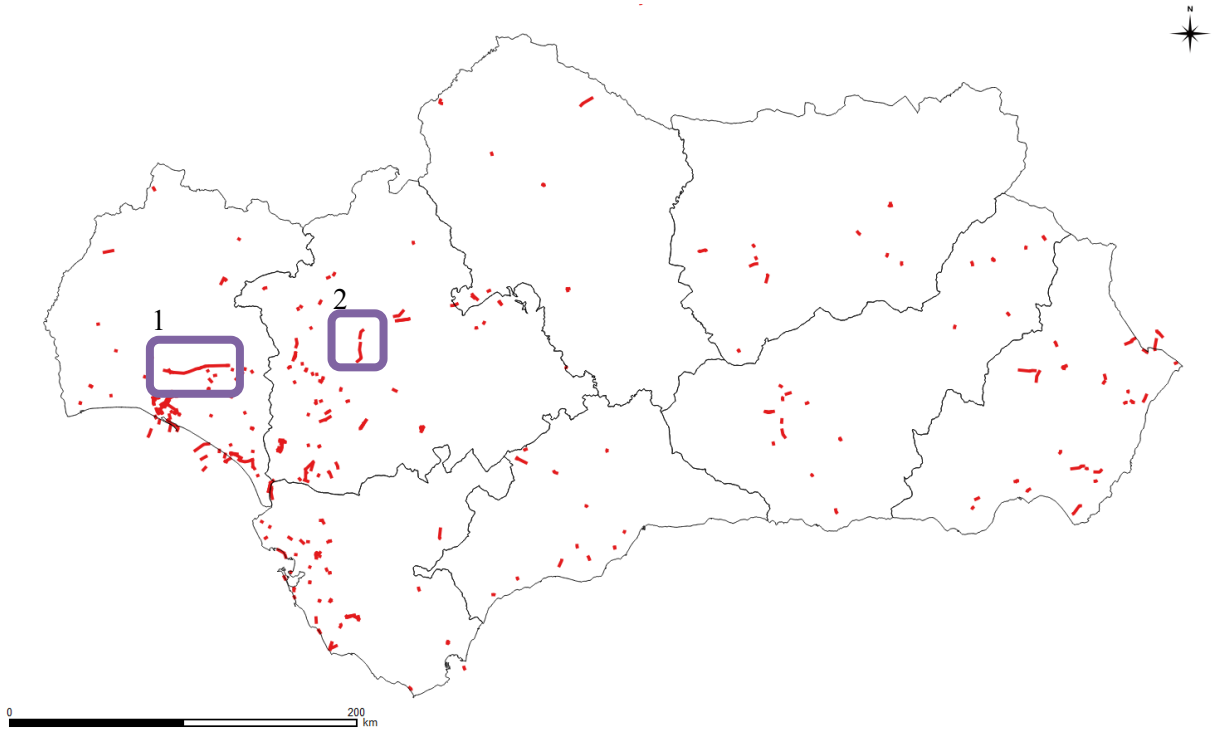


Figura 4-20. Rutas 2012

Figura 4-21. Ruta 1_2012

Tabla 4-7- Ruta 1_2012

Año	**2012**
Nº Ruta	1
Observaciones	Posibles actuaciones en línea eléctrica.

Figura 4-22. Ruta 2_2012

Tabla 4-8. Ruta 2_2012

Año	**2012**
Nº Ruta	2
Observaciones	Caminos en los alrededores del aeropuerto de Sevilla.

4.3.6 Rutas "nuevas" 2013

En la Figura 4-23 se representa la ruta más destacada de este año.

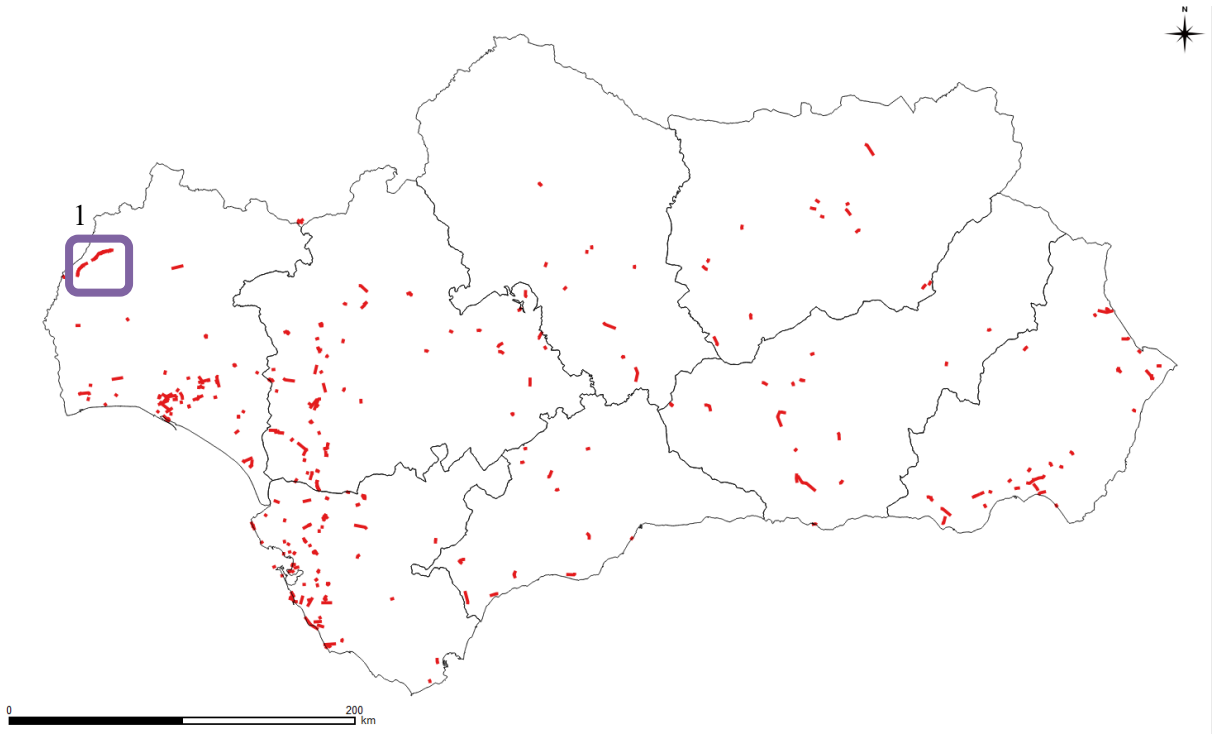


Figura 4-23. Rutas 2013

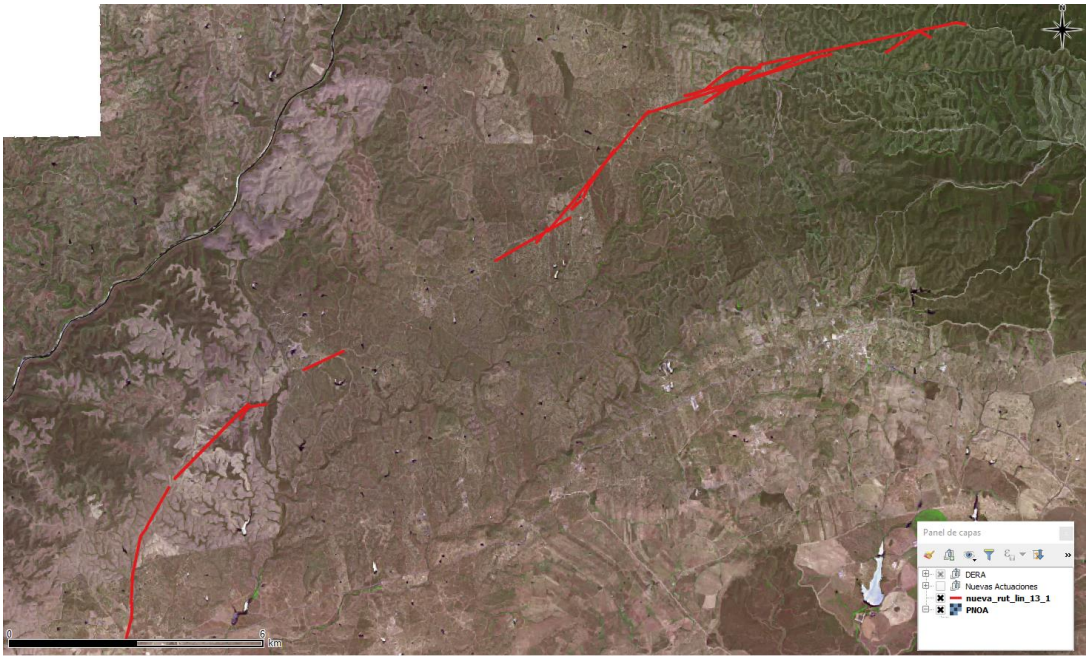


Figura 4-24. Ruta 1_2013

Tabla 4-9. Ruta 1_2013

Año	**2013**
Nº Ruta	1
Observaciones	En este caso se trata del camino de Paymoga a cañada de la frontera

4.3.7 Rutas "nuevas" 2014

La Figura 4-25 representa la ubicación de la ruta a estudiar.

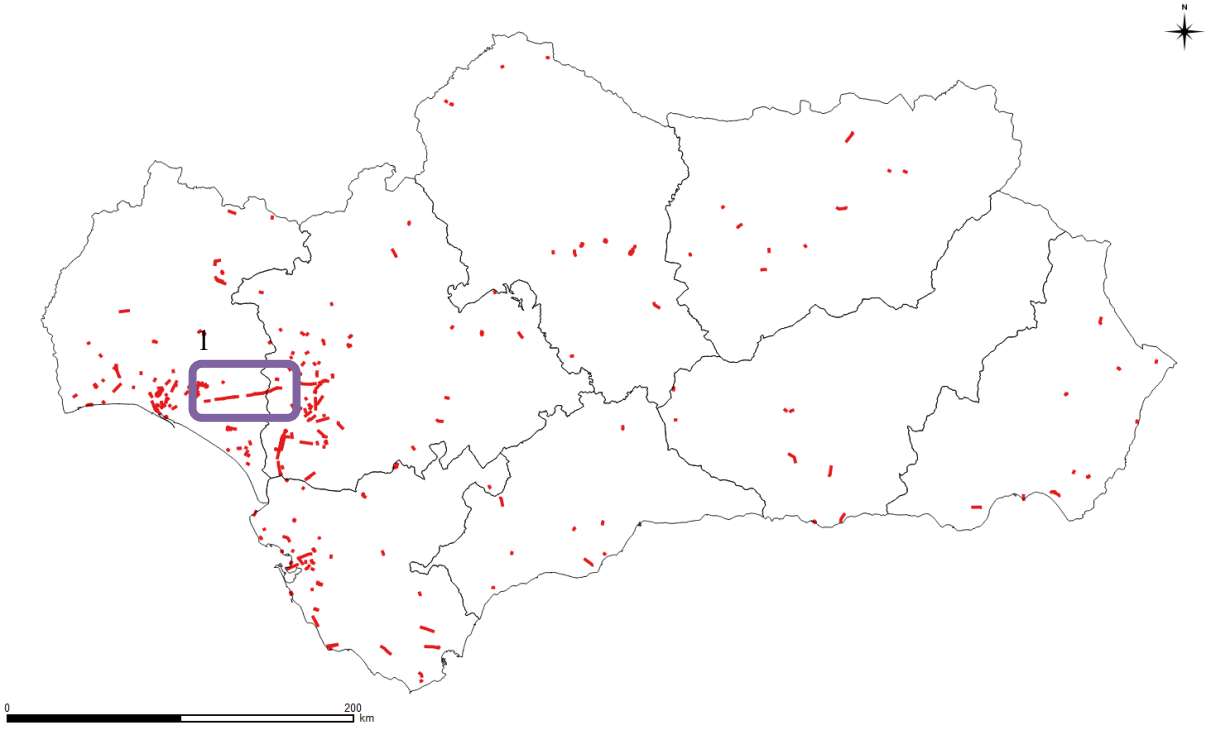


Figura 4-25. Rutas 2014

Figura 4-26. Ruta 1_2014

Tabla 4-10. Ruta 1_2014

Año	**2014**
Nº Ruta	1
Observaciones	Posible trazado de un gasoducto

4.3.8 Rutas “nuevas” 2015

En la Figura 4-27 aparecen la ubicación de la obra destacada en este año.

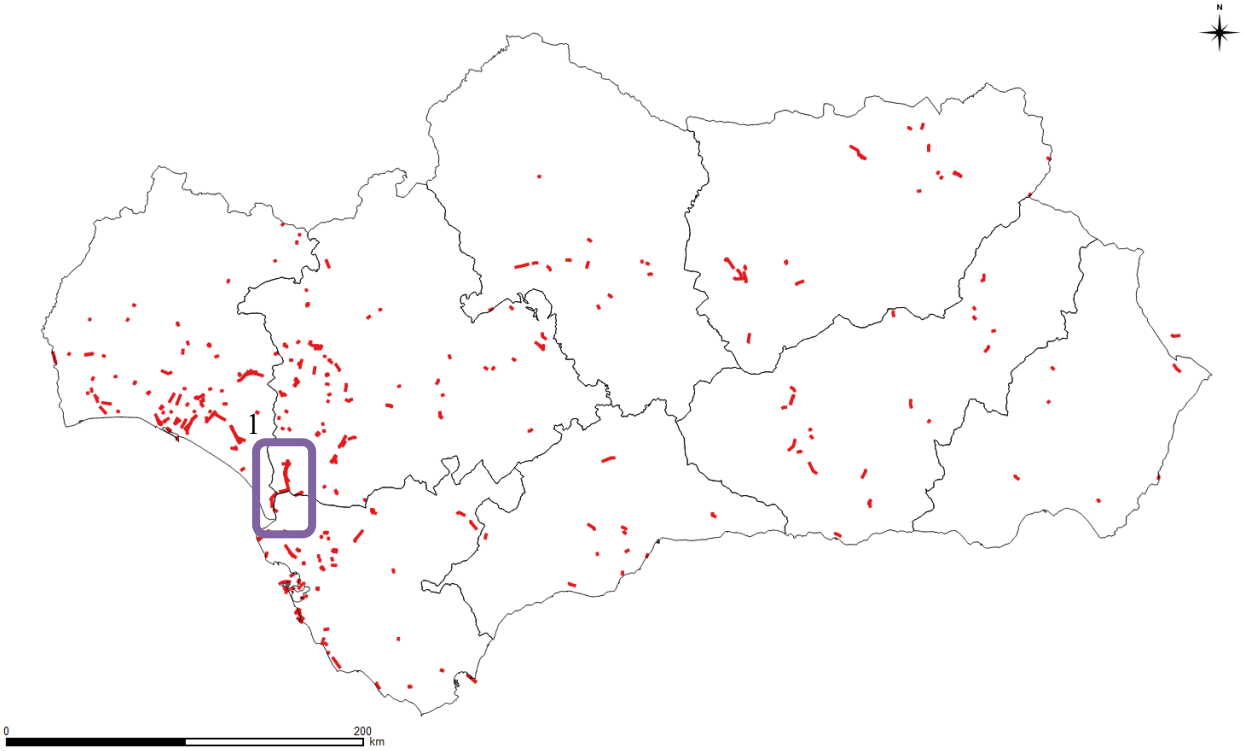


Figura 4-27. Rutas 2015

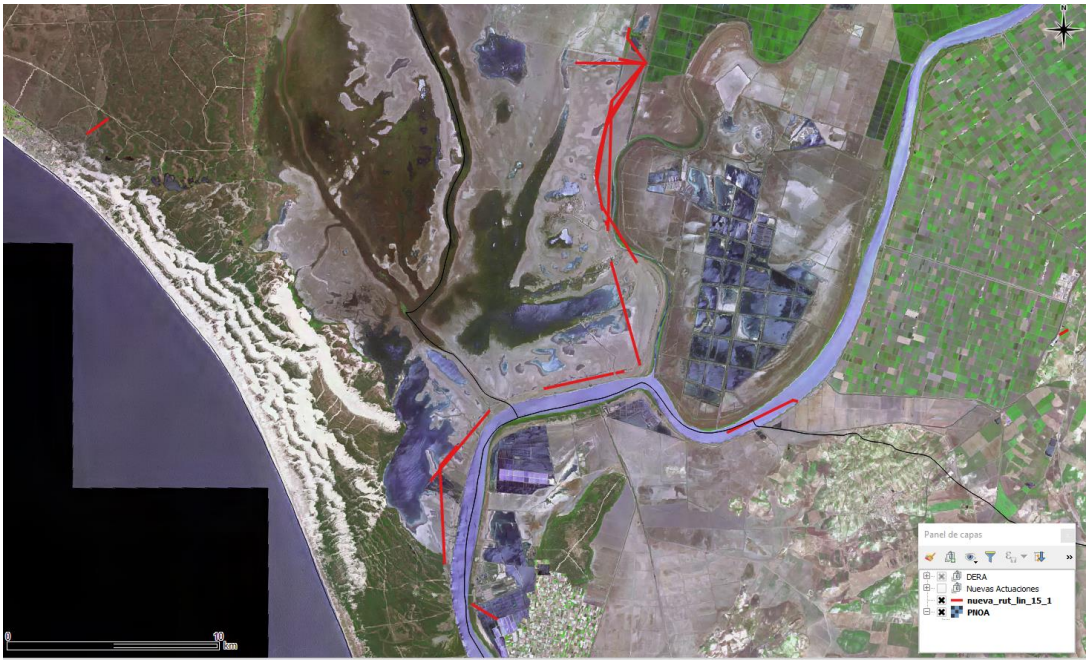


Figura 4-28. Ruta 1_2015

Tabla 4-11. Ruta 1_2015

Año	**2015**
Nº Ruta	1
Observaciones	Actuaciones en la zona del delta del Guadalquivir

4.3.9 Rutas “nuevas” 2016

En la Figura 4-29 se resalta la ruta que se va a analizar en el año 2016.

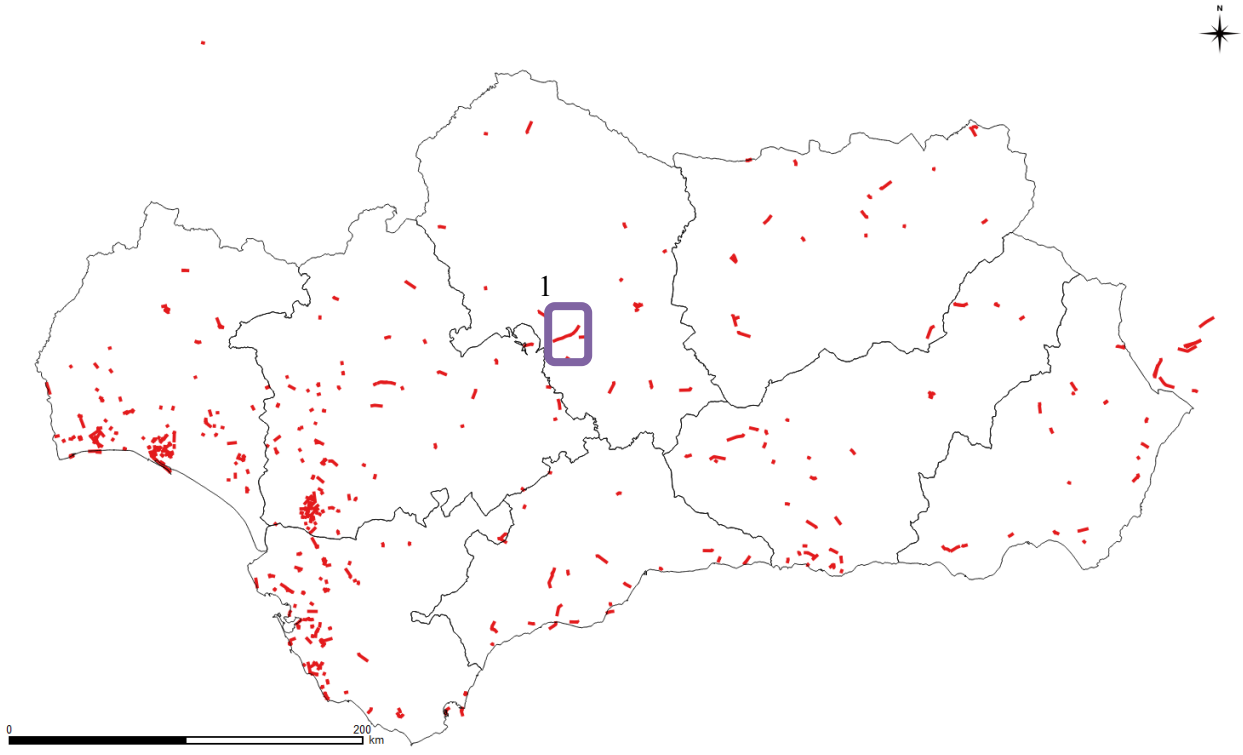


Figura 4-29. Rutas 2016

Figura 4-30. Ruta 1_2016

Tabla 4-12. Ruta 1_2016

Año	**2016**
Nº Ruta	1
Observaciones	Posible trazado de un tramo del gasoducto Sevilla - Córdoba

5 DISCUSIONES/CONCLUSIONES

En este apartado se presentará un análisis mediante el cual se buscaba generan un algoritmo capaz de identificar el tipo de obra lineal identificada en cada caso. Una vez presentado, se concluirán las razones por las cuales no será de utilidad a priori

Por último, en este capítulo se discutirá sobre la utilidad de toda la metodología presentada en el proyecto. Así mismo, se propondrán posibles vías para mejorar o evolucionar el estudio, de cara a ser más preciso y por tanto útil.

5.1 Algoritmo categorizador

La creación de este algoritmo irá orientada a la identificación de nuevas obras lineales, atendiendo a criterios de comportamiento del topógrafo. Para esta tarea, el código final se debería basar en un modelo de predicción tipo árbol de decisión (*desition tree*). Esta técnica de *Machine Learning*, se basa en establecer una serie de reglas anidadas que permitan clasificar los elementos de entrada según cumplan o no dichas reglas [6].

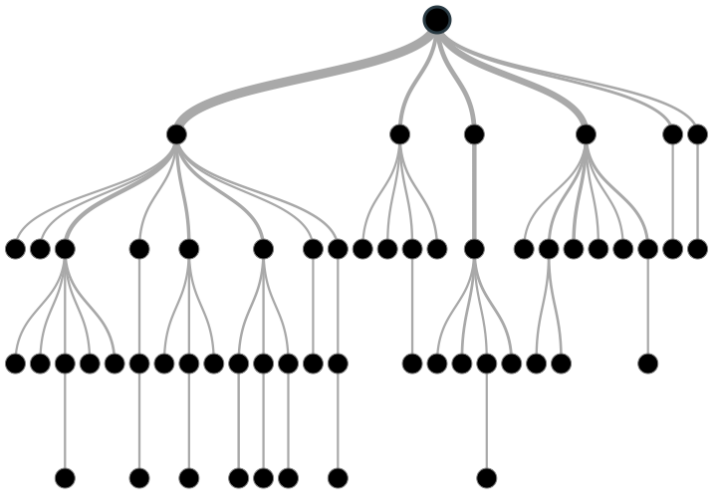


Figura 5-1. Árbol de decisión

En este proyecto, los parámetros de entrada serán las conexiones, o más concretamente, las rutas no categorizadas obtenidas mediante los métodos anteriores. Estas rutas dependen de las conexiones originales, cuyos atributos atenderán a criterios temporales o espaciales. Es por esto por lo que esas dos magnitudes serán las únicas que se podrían utilizar a la hora de elaborar las reglas.

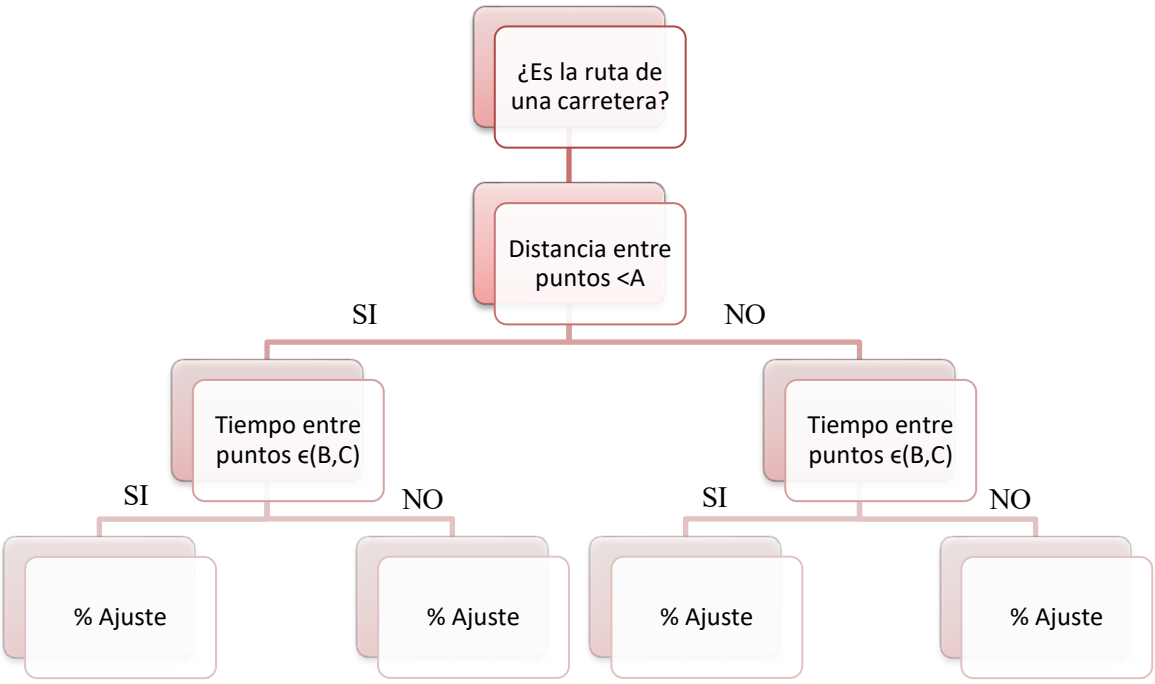


Figura 5-2. Árbol con reglas tiempo/espacio. Ejemplo

En la Figura 5-2 se muestra un ejemplo de cómo podría estar estructurado el árbol de decisión correspondiente a los datos de un año cualquiera. Para que este ejemplo pasara a ser el esquema real de funcionamiento habría que encontrar los valores de tiempo y espacio (distancia o curvatura, por ejemplo) que permitan establecer reglas que diferencien entre los distintos tipos de obras posibles.

A tenor de los datos contenidos en los campos de atributos de las conexiones de partida, la búsqueda de estos valores se traducirá en:

- Búsqueda de relaciones entre fecha/hora de un punto y el siguiente.
- Búsqueda de relaciones entre distancias entre puntos.

Un estudio de los campos de tiempo de las diferentes capas de puntos mostrará que será casi imposible encontrar relaciones temporales entre puntos de una misma obra. Esto se deberá a las siguientes razones:

- La diferencia de tiempo entre el fin de la conexión de un punto y el siguiente (perteneciente a la misma ruta) suele ser muy bajo y de valor muy similar ya sea carretera, ferrocarril o cualquier otra categoría.
- En muchos casos, debido al fin de la jornada laboral, el punto siguiente es tomado el día posterior, o incluso varios días después.
- Muchas rutas vienen dadas por puntos tomados según un orden azaroso, pudiendo haber 30 segundos entre algunos puntos y horas entre otros (perteneciendo todos a la misma obra).
- El campo “DURATION” que indica la duración de la conexión, tampoco nos sirve como identificador puesto que su valor varía independientemente del tipo de obra lineal.

Para estudiar ciertos patrones entre las distancias de toma de puntos, además del estudio previo, se genera un algoritmo que trate de encontrar alguna tendencia. Este algoritmo estará descrito en el siguiente apartado.

5.1.1 Análisis de distancias

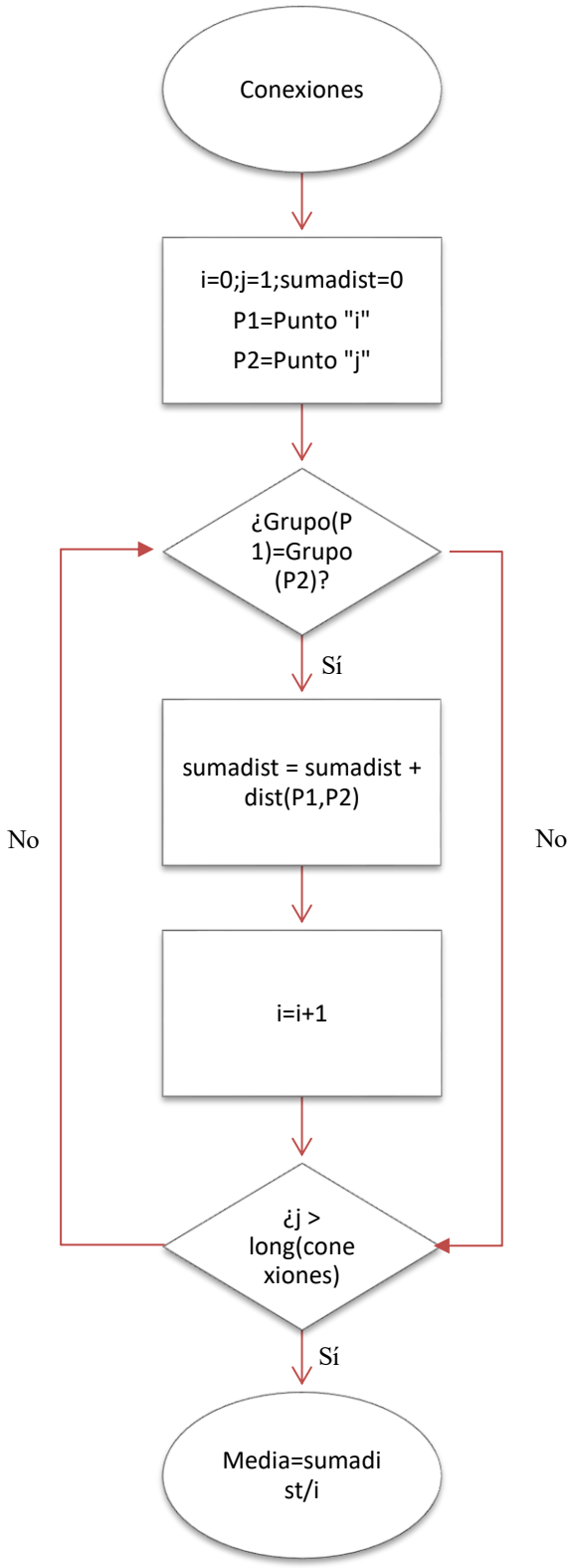


Figura 5-3. Algoritmo análisis de distancias

Como vemos en la Figura 5-3, mediante este código (que también estará incluido en el anexo) se tratará de encontrar cierta tendencia en cada una de las categorías de obras lineales. Para ello, se aplicará el algoritmo a las conexiones categorizadas en capítulos anteriores, obteniendo de esta forma estadísticas para cada caso.

Tras la aplicación del algoritmo, los datos obtenidos expresados en kilómetros son los siguientes:

Tabla 5-1. Distancias por categoría-año

Categoría	2008	2009	2010	2011	2012	2013	2014	2015	2016	Media Global
Carreteras	0.485	0.603	0.651	0.613	0.469	0.673	0.562	0.747	0.928	0.637
FFCC	0.560	0.438	0.595	0.624	0.392	0.386	0.498	0.644	0.503	0.516
Conducciones	0.373	0.567	0.462	0.442	0.383	0.432	0.619	0.750	0.640	0.519
Líneas Eléctricas	0.205	0.346	0.489	0.321	0.331	0.339	0.263	0.491	0.309	0.344
Gasoductos	0.581	-	-	-	-	0.581	0.730	0.926	0.950	0.800

Como se refleja en la Tabla 5-1, aunque los puntos presentan ciertas tendencias en algunos casos, los valores estarán demasiado próximos como para tomarlos como reglas capaces de identificar nuevas actuaciones. Además, dentro de sus valores similares existe demasiada variabilidad entre las conexiones de un mismo tipo para distintos años, lo cual dificultaría aún más la tarea de identificación.

5.2 Conclusiones

Con todo lo expuesto en este proyecto se puede llegar a las siguientes conclusiones:

- La identificación de patrones lineales mediante programación es posible, sin embargo, se puede ganar mucho en precisión mediante técnicas más avanzadas y/o mejorando en el tratamiento y obtención del dato de partida.
- Aún con esta identificación, se hace muy complicada la diferenciación entre verdaderas obras y otro tipo de patrones.
- La clasificación de posibles obras lineales mediante un algoritmo informático dependerá de la calidad y cantidad de datos de partida, pues cuanto mayor sea, más se eliminará el factor humano del topógrafo que hace que la variabilidad entre patrones de un mismo tipo sea muy alta.

ANEXO I.- CÓDIGOS

Para algunos de los códigos contenidos en este anexo, habrá que tener en cuenta que se muestran aquellos válidos para las conexiones del año 2016. Para cualquier otro año habría se emplean los mismos códigos, pero cambiando el año en cuestión.

limpiezapuntos.py

```
1. ##limpieza=name
2. ##nucleos=vector
3. ##conexiones=vector
4. outputs_QGISFIXEDDISTANCEBUFFER_1=processing.runalg('qgis:fixeddistancebuffer', nucleos,100.0,5.0,False,N
   one)
5. outputs_QGISEXTRACTBYLOCATION_1=processing.runalg('qgis:extractbylocation', conexiones,outputs_QGISFIX
   EDDISTANCEBUFFER_1['OUTPUT'],['disjoint'],0.0,None)
6. outputs_QGISFIXEDDISTANCEBUFFER_2=processing.runalg('qgis:fixeddistancebuffer', outputs_QGISEXTRACTB
   YLOCATION_1['OUTPUT'],5000.0,5.0,False,None)
7. outputs_QGISCOUNTPOINTSINPOLYGON_1=processing.runalg('qgis:countpointsinpolygon', outputs_QGISFIXED
   DISTANCEBUFFER_2['OUTPUT'],conexiones,'NUMPOINTS',None)
8. outputs_QGISEXTRACTBYATTRIBUTE_1=processing.runalg('qgis:extractbyattribute', outputs_QGISCOUNTPOINT
   SINPOLYGON_1['OUTPUT'],'NUMPOINTS',0,'1',None)
9. outputs_QGISEXTRACTBYLOCATION_2=processing.runalg('qgis:extractbylocation', outputs_QGISEXTRACTBYLO
   CATION_1['OUTPUT'],outputs_QGISEXTRACTBYATTRIBUTE_1['OUTPUT'],['disjoint'],0.0,None)
```

lin_pattern_16.py

```
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_16_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
   attrs = { 0 : 1}
   layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
   last_id=feature.id()

grupo=1
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
   counter=counter+1
   counter_2=counter+1
   if counter_2>last_id:
      break

   fid = counter # the numbered feature (zero based indexing!)
   iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
   feature = next(iterator)
   attrs = feature.attributes()

   fid_2 = counter_2 # the numbered feature (zero based indexing!)
   iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
   feature_2 = next(iterator_2)
   attrs_2 = feature_2.attributes()

   coordx_1=attrs[-3]
   coordy_1=attrs[-2]
   coordx_2=attrs_2[-3]
   coordy_2=attrs_2[-2]
   dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


   if attrs[1]!=attrs_2[1] or dist>5000 or dist<100:
      #print 'iguales'
      #print grupo
   #else:
      grupo=grupo+1

   #print counter
   #print counter_2

   #Add grupo a la capa 1
   if caps & QgsVectorDataProvider.ChangeAttributeValues:
      attrs2 = { 14 : grupo}
      layer.dataProvider().changeAttributeValues({ fid_2 : attrs2 })
   layer.updateFields()



#Puntos a ruta
import processing
```

```
64. processing.runalg('qgis:pointstopath', layer,'Grupo','Id',None,'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/rut_lin_16.shp',None)
65.
66. #Abrimos la capa creada
67. layer_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/rut_lin_16.shp",'rut_lin_16',"ogr")
68. QgsMapLayerRegistry.instance().addMapLayer(layer_lin)
```

sinuosidad.py

```
1.  import processing
2.
3.  lines = QgsMapLayerRegistry.instance().mapLayersByName('rut_lin_16')[0]
4.
5.  processing.runandload('qgis:fieldcalculator',
6.                  {"INPUT_LAYER": lines,
7.                  "FIELD_NAME": "sinuosity",
8.                  "FIELD_TYPE": 0,
9.                  "FORMULA": '$length / sqrt((xat(-1) - xat(0))^2 + (yat(-1) - yat(0))^2)',
10.                 "NEW_FIELD": True,
11.                 "OUTPUT_LAYER": None})
12.
13. lines2 = QgsMapLayerRegistry.instance().mapLayersByName('Calculado')[0]
14.
15. processing.runalg('qgis:extractbyexpression',
16.                   {"INPUT": lines2,
17.                  "EXPRESSION": '"sinuosity"!=1 AND "sinuosity"<1.1',
18.                  "OUTPUT": 'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_SINUOSO/rut_lin_16_1.shp
    '})
19.
20. #Abrimos la capa creada
21. capa_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_SINUOSO/rut_lin_16_
    1.shp",'rut_lin_16_1',"ogr")
22. QgsMapLayerRegistry.instance().addMapLayer(capa_lin)
23.
24. #Borrar Calculado
25. QgsMapLayerRegistry.instance().removeMapLayer(lines2)
```

categoriza.py

```
1. ##categorizador=name
2. ##categoria=vector
3. ##conexion=vector
4. ##conexcat=output vector
5. outputs_QGISFIXEDDISTANCEBUFFER_1=processing.runalg('qgis:fixeddistancebuffer', categoria,50.0,5.0,False,None)
6. outputs_QGISEXTRACTBYLOCATION_1=processing.runalg('qgis:extractbylocation', conexion,outputs_QGISFIXEDDISTANCEBUFFER_1['OUTPUT'],['within'],0.0,conexcat)
```

carret_16.py

```
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_16_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
   attrs = { 0 : 1}
   layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
   last_id=feature.id()

grupo=1
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
   counter=counter+1
   counter_2=counter+1
   if counter_2>last_id:
      break

   fid = counter # the numbered feature (zero based indexing!)
   iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
   feature = next(iterator)
   attrs = feature.attributes()

   fid_2 = counter_2 # the numbered feature (zero based indexing!)
   iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
   feature_2 = next(iterator_2)
   attrs_2 = feature_2.attributes()

   coordx_1=attrs[-3]
   coordy_1=attrs[-2]
   coordx_2=attrs_2[-3]
   coordy_2=attrs_2[-2]
   dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


   if attrs[1]!=attrs_2[1] or dist>7000:
      #print 'iguales'
      #print grupo
   #else:
      grupo=grupo+1

   #print counter
   #print counter_2

   #Add grupo a la capa 1
   if caps & QgsVectorDataProvider.ChangeAttributeValues:
      attrs2 = { 14 : grupo}
      layer.dataProvider().changeAttributeValues({ fid_2 : attrs2 })
   layer.updateFields()



#Puntos a ruta
import processing
```

```
processing.runalg('qgis:pointstopath', layer,'Grupo','Id',None,'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_carret.shp',None)

#Abrimos la capa creada
layer_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_carret.shp",'rut_lin_16_carret',"ogr")
QgsMapLayerRegistry.instance().addMapLayer(layer_lin)
```

cond_16.py

```
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_16_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
   attrs = { 0 : 1}
   layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
   last_id=feature.id()

grupo=1
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
   counter=counter+1
   counter_2=counter+1
   if counter_2>last_id:
      break

   fid = counter # the numbered feature (zero based indexing!)
   iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
   feature = next(iterator)
   attrs = feature.attributes()

   fid_2 = counter_2 # the numbered feature (zero based indexing!)
   iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
   feature_2 = next(iterator_2)
   attrs_2 = feature_2.attributes()

   coordx_1=attrs[-3]
   coordy_1=attrs[-2]
   coordx_2=attrs_2[-3]
   coordy_2=attrs_2[-2]
   dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


   if attrs[1]!=attrs_2[1] or dist>7000:
      #print 'iguales'
      #print grupo
   #else:
      grupo=grupo+1

   #print counter
   #print counter_2

   #Add grupo a la capa 1
   if caps & QgsVectorDataProvider.ChangeAttributeValues:
      attrs2 = { 14 : grupo}
      layer.dataProvider().changeAttributeValues({ fid_2 : attrs2 })
   layer.updateFields()



#Puntos a ruta
import processing
```

```
64. processing.runalg('qgis:pointstopath', layer,'Grupo','Id',None,'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_cond.shp',None)
65.
66. #Abrimos la capa creada
67. layer_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_cond.shp",'rut_lin_16_cond',"ogr")
68. QgsMapLayerRegistry.instance().addMapLayer(layer_lin)
```

elect_16.py

```
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_16_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
   attrs = { 0 : 1}
   layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
   last_id=feature.id()

grupo=1
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
   counter=counter+1
   counter_2=counter+1
   if counter_2>last_id:
      break

   fid = counter # the numbered feature (zero based indexing!)
   iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
   feature = next(iterator)
   attrs = feature.attributes()

   fid_2 = counter_2 # the numbered feature (zero based indexing!)
   iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
   feature_2 = next(iterator_2)
   attrs_2 = feature_2.attributes()

   coordx_1=attrs[-3]
   coordy_1=attrs[-2]
   coordx_2=attrs_2[-3]
   coordy_2=attrs_2[-2]
   dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


   if attrs[1]!=attrs_2[1] or dist>7000:
      #print 'iguales'
      #print grupo
   #else:
      grupo=grupo+1

   #print counter
   #print counter_2

   #Add grupo a la capa 1
   if caps & QgsVectorDataProvider.ChangeAttributeValues:
      attrs2 = { 14 : grupo}
      layer.dataProvider().changeAttributeValues({ fid_2 : attrs2 })
   layer.updateFields()



#Puntos a ruta
```

```
63. import processing
64. processing.runalg('qgis:pointstopath', layer,'Grupo','Id',None,'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_elect.shp',None)
65.
66. #Abrimos la capa creada
67. layer_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_elect.shp",'rut_lin_16_elect',"ogr")
68. QgsMapLayerRegistry.instance().addMapLayer(layer_lin)
```

ffcc_16.py

```
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_16_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
   attrs = { 0 : 1}
   layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
   last_id=feature.id()

grupo=1
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
   counter=counter+1
   counter_2=counter+1
   if counter_2>last_id:
      break

   fid = counter # the numbered feature (zero based indexing!)
   iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
   feature = next(iterator)
   attrs = feature.attributes()

   fid_2 = counter_2 # the numbered feature (zero based indexing!)
   iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
   feature_2 = next(iterator_2)
   attrs_2 = feature_2.attributes()

   coordx_1=attrs[-3]
   coordy_1=attrs[-2]
   coordx_2=attrs_2[-3]
   coordy_2=attrs_2[-2]
   dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


   if attrs[1]!=attrs_2[1] or dist>7000:
      #print 'iguales'
      #print grupo
   #else:
      grupo=grupo+1

   #print counter
   #print counter_2

   #Add grupo a la capa 1
   if caps & QgsVectorDataProvider.ChangeAttributeValues:
      attrs2 = { 14 : grupo}
      layer.dataProvider().changeAttributeValues({ fid_2 : attrs2 })
   layer.updateFields()



#Puntos a ruta
```

```
63. import processing
64. processing.runalg('qgis:pointstopath', layer,'Grupo','Id',None,'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_ffcc.shp',None)
65.
66. #Abrimos la capa creada
67. layer_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_ffcc.shp",'rut_lin_16_ffcc',"ogr")
68. QgsMapLayerRegistry.instance().addMapLayer(layer_lin)
```

gas_16.py

```
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('gas_conex_16')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
   attrs = { 0 : 1}
   layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
   last_id=feature.id()

grupo=1
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
   counter=counter+1
   counter_2=counter+1
   if counter_2>last_id:
      break

   fid = counter # the numbered feature (zero based indexing!)
   iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
   feature = next(iterator)
   attrs = feature.attributes()

   fid_2 = counter_2 # the numbered feature (zero based indexing!)
   iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
   feature_2 = next(iterator_2)
   attrs_2 = feature_2.attributes()

   coordx_1=attrs[-3]
   coordy_1=attrs[-2]
   coordx_2=attrs_2[-3]
   coordy_2=attrs_2[-2]
   dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


   if attrs[1]!=attrs_2[1] or dist>7000:
      #print 'iguales'
      #print grupo
   #else:
      grupo=grupo+1

   #print counter
   #print counter_2

   #Add grupo a la capa 1
   if caps & QgsVectorDataProvider.ChangeAttributeValues:
      attrs2 = { 14 : grupo}
      layer.dataProvider().changeAttributeValues({ fid_2 : attrs2 })
   layer.updateFields()



#Puntos a ruta
```

```
import processing
processing.runalg('qgis:pointstopath', layer,'Grupo','Id',None,'C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_gas.shp',None)

#Abrimos la capa creada
layer_lin = QgsVectorLayer(r"C:/Users/josea/Documents/JOSEAPH/ETSI/TFG/FIXED/SHP_LIN_PATT/categ/rut_lin_16_gas.shp",'rut_lin_16_gas',"ogr")
QgsMapLayerRegistry.instance().addMapLayer(layer_lin)
```

analiza_pattern_carret.py

```
#2008

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_08_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm08=mediaDist/1000
print 'Media dist08 en km'
print distsumkm08


#2009

#Agrupar puntos por campos
```

```
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_09_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm09=mediaDist/1000
print 'Media dist09 en km'
print distsumkm09
#2010

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_10_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
```

```
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm10=mediaDist/1000
print 'Media dist10 en km'
print distsumkm10
#2011

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_11_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
```

```
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm11=mediaDist/1000
print 'Media dist11 en km'
print distsumkm11
#2012

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_12_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

```

```
    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm12=mediaDist/1000
print 'Media dist12 en km'
print distsumkm12
#2013

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_13_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

```

```
    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm13=mediaDist/1000
print 'Media dist13 en km'
print distsumkm13
#2014

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_14_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
```

```
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm14=mediaDist/1000
print 'Media dist14 en km'
print distsumkm14
#2015

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_15_carret')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm15=mediaDist/1000
print 'Media dist15 en km'
print distsumkm15
#2016
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify
```

```
462.
463.layer= QgsMapLayerRegistry.instance().mapLayersByName('conex_16_carret')[0]
464.caps = layer.dataProvider().capabilities()
465.
466.#ponemos 1 en primera fila
467.if caps & QgsVectorDataProvider.ChangeAttributeValues:
468.    attrs = { 0 : 1}
469.    layer.dataProvider().changeAttributeValues({ fid : attrs })
470.layer.updateFields()
471.
472.iter = layer.getFeatures()
473.for feature in iter:
474.    last_id=feature.id()
475.
476.
477.
478.distsum=0
479.Nsum=0
480.counter=-1
481.#Para tomar el ultimo ID
482.iter = layer.getFeatures()
483.for feature in iter:
484.    counter=counter+1
485.    counter_2=counter+1
486.    if counter_2>last_id:
487.        break
488.
489.    fid = counter # the numbered feature (zero based indexing!)
490.    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
491.    feature = next(iterator)
492.    attrs = feature.attributes()
493.
494.    fid_2 = counter_2 # the numbered feature (zero based indexing!)
495.    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
496.    feature_2 = next(iterator_2)
497.    attrs_2 = feature_2.attributes()
498.
499.    coordx_1=attrs[-3]
500.    coordy_1=attrs[-2]
501.    coordx_2=attrs_2[-3]
502.    coordy_2=attrs_2[-2]
503.    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5
504.
505.
506.    if attrs[-1]==attrs_2[-1]:
507.        distsum=dist+distsum
508.        Nsum=Nsum+1
509.
510.
511.mediaDist=distsum/Nsum
512.distsumkm16=mediaDist/1000
513.print 'Media dist16 en km'
514.print distsumkm16
515.mediadist=(distsumkm08+distsumkm09+distsumkm10+distsumkm11+distsumkm12+distsumkm13+dist
    sumkm14+distsumkm15+distsumkm16)/9
516.print 'Media dist en km de carreteras'
517.print mediadist
```

analiza_pattern_cond.py

```
#2008

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_08_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm08=mediaDist/1000
print 'Media dist08 en km'
print distsumkm08


#2009

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

```

```
layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_09_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm09=mediaDist/1000
print 'Media dist09 en km'
print distsumkm09
#2010

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_10_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

```

```
iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm10=mediaDist/1000
print 'Media dist10 en km'
print distsumkm10
#2011

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_11_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
```

```
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm11=mediaDist/1000
print 'Media dist11 en km'
print distsumkm11
#2012

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_12_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
```

```
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm12=mediaDist/1000
print 'Media dist12 en km'
print distsumkm12
#2013

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_13_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
```

```
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm13=mediaDist/1000
print 'Media dist13 en km'
print distsumkm13
#2014

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_14_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1

```

```
397.
398.mediaDist=distsum/Nsum
399.distsumkm14=mediaDist/1000
400.print 'Media dist14 en km'
401.print distsumkm14
402.#2015
403.
404.#Agrupar puntos por campos
405.fid = 0   # ID of the feature we will modify
406.
407.layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_15_2')[0]
408.caps = layer.dataProvider().capabilities()
409.
410.#ponemos 1 en primera fila
411.if caps & QgsVectorDataProvider.ChangeAttributeValues:
412.    attrs = { 0 : 1}
413.    layer.dataProvider().changeAttributeValues({ fid : attrs })
414.layer.updateFields()
415.
416.iter = layer.getFeatures()
417.for feature in iter:
418.    last_id=feature.id()
419.
420.
421.
422.distsum=0
423.Nsum=0
424.counter=-1
425.#Para tomar el ultimo ID
426.iter = layer.getFeatures()
427.for feature in iter:
428.    counter=counter+1
429.    counter_2=counter+1
430.    if counter_2>last_id:
431.        break
432.
433.    fid = counter # the numbered feature (zero based indexing!)
434.    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
435.    feature = next(iterator)
436.    attrs = feature.attributes()
437.
438.    fid_2 = counter_2 # the numbered feature (zero based indexing!)
439.    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
440.    feature_2 = next(iterator_2)
441.    attrs_2 = feature_2.attributes()
442.
443.    coordx_1=attrs[-3]
444.    coordy_1=attrs[-2]
445.    coordx_2=attrs_2[-3]
446.    coordy_2=attrs_2[-2]
447.    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5
448.
449.
450.    if attrs[-1]==attrs_2[-1]:
451.        distsum=dist+distsum
452.        Nsum=Nsum+1
453.
454.
455.mediaDist=distsum/Nsum
456.distsumkm15=mediaDist/1000
457.print 'Media dist15 en km'
458.print distsumkm15
459.#2016
460.#Agrupar puntos por campos
461.fid = 0   # ID of the feature we will modify
462.
463.layer= QgsMapLayerRegistry.instance().mapLayersByName('cond_conex_16_2')[0]
```

```
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm16=mediaDist/1000
print 'Media dist16 en km'
print distsumkm16
mediadist=(distsumkm08+distsumkm09+distsumkm10+distsumkm11+distsumkm12+distsumkm13+dist
sumkm14+distsumkm15+distsumkm16)/9
print 'Media dist en km de conducciones'
print mediadist
```

analiza_pattern_elect.py

```
#2008

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_08_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm08=mediaDist/1000
print 'Media dist08 en km'
print distsumkm08


#2009

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

```

```
layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_09_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm09=mediaDist/1000
print 'Media dist09 en km'
print distsumkm09
#2010

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_10_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

```

```
iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm10=mediaDist/1000
print 'Media dist10 en km'
print distsumkm10
#2011

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_11_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
```

```
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm11=mediaDist/1000
print 'Media dist11 en km'
print distsumkm11
#2012

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_12_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
```

```
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm12=mediaDist/1000
print 'Media dist12 en km'
print distsumkm12
#2013

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_13_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
```

```
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm13=mediaDist/1000
print 'Media dist13 en km'
print distsumkm13
#2014

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_14_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1

```

```
397.
398.mediaDist=distsum/Nsum
399.distsumkm14=mediaDist/1000
400.print 'Media dist14 en km'
401.print distsumkm14
402.#2015
403.
404.#Agrupar puntos por campos
405.fid = 0   # ID of the feature we will modify
406.
407.layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_15_2')[0]
408.caps = layer.dataProvider().capabilities()
409.
410.#ponemos 1 en primera fila
411.if caps & QgsVectorDataProvider.ChangeAttributeValues:
412.    attrs = { 0 : 1}
413.    layer.dataProvider().changeAttributeValues({ fid : attrs })
414.layer.updateFields()
415.
416.iter = layer.getFeatures()
417.for feature in iter:
418.    last_id=feature.id()
419.
420.
421.
422.distsum=0
423.Nsum=0
424.counter=-1
425.#Para tomar el ultimo ID
426.iter = layer.getFeatures()
427.for feature in iter:
428.    counter=counter+1
429.    counter_2=counter+1
430.    if counter_2>last_id:
431.        break
432.
433.    fid = counter # the numbered feature (zero based indexing!)
434.    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
435.    feature = next(iterator)
436.    attrs = feature.attributes()
437.
438.    fid_2 = counter_2 # the numbered feature (zero based indexing!)
439.    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
440.    feature_2 = next(iterator_2)
441.    attrs_2 = feature_2.attributes()
442.
443.    coordx_1=attrs[-3]
444.    coordy_1=attrs[-2]
445.    coordx_2=attrs_2[-3]
446.    coordy_2=attrs_2[-2]
447.    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5
448.
449.
450.    if attrs[-1]==attrs_2[-1]:
451.        distsum=dist+distsum
452.        Nsum=Nsum+1
453.
454.
455.mediaDist=distsum/Nsum
456.distsumkm15=mediaDist/1000
457.print 'Media dist15 en km'
458.print distsumkm15
459.#2016
460.#Agrupar puntos por campos
461.fid = 0   # ID of the feature we will modify
462.
463.layer= QgsMapLayerRegistry.instance().mapLayersByName('elect_conex_16_2')[0]
```

```
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm16=mediaDist/1000
print 'Media dist16 en km'
print distsumkm16
mediadist=(distsumkm08+distsumkm09+distsumkm10+distsumkm11+distsumkm12+distsumkm13+dist
sumkm14+distsumkm15+distsumkm16)/9
print 'Media dist en km de lineas electricas'
print mediadist
```

analiza_pattern_ffcc.py

```
#2008

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_08_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm08=mediaDist/1000
print 'Media dist08 en km'
print distsumkm08


#2009

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

```

```
layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_09_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm09=mediaDist/1000
print 'Media dist09 en km'
print distsumkm09
#2010

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_10_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

```

```
iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm10=mediaDist/1000
print 'Media dist10 en km'
print distsumkm10
#2011

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_11_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
```

```
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm11=mediaDist/1000
print 'Media dist11 en km'
print distsumkm11
#2012

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_12_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
```

```
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm12=mediaDist/1000
print 'Media dist12 en km'
print distsumkm12
#2013

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_13_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
```

```
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm13=mediaDist/1000
print 'Media dist13 en km'
print distsumkm13
#2014

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_14_2')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1

```

```
397.
398.mediaDist=distsum/Nsum
399.distsumkm14=mediaDist/1000
400.print 'Media dist14 en km'
401.print distsumkm14
402.#2015
403.
404.#Agrupar puntos por campos
405.fid = 0   # ID of the feature we will modify
406.
407.layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_15_2')[0]
408.caps = layer.dataProvider().capabilities()
409.
410.#ponemos 1 en primera fila
411.if caps & QgsVectorDataProvider.ChangeAttributeValues:
412.    attrs = { 0 : 1}
413.    layer.dataProvider().changeAttributeValues({ fid : attrs })
414.layer.updateFields()
415.
416.iter = layer.getFeatures()
417.for feature in iter:
418.    last_id=feature.id()
419.
420.
421.
422.distsum=0
423.Nsum=0
424.counter=-1
425.#Para tomar el ultimo ID
426.iter = layer.getFeatures()
427.for feature in iter:
428.    counter=counter+1
429.    counter_2=counter+1
430.    if counter_2>last_id:
431.        break
432.
433.    fid = counter # the numbered feature (zero based indexing!)
434.    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
435.    feature = next(iterator)
436.    attrs = feature.attributes()
437.
438.    fid_2 = counter_2 # the numbered feature (zero based indexing!)
439.    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
440.    feature_2 = next(iterator_2)
441.    attrs_2 = feature_2.attributes()
442.
443.    coordx_1=attrs[-3]
444.    coordy_1=attrs[-2]
445.    coordx_2=attrs_2[-3]
446.    coordy_2=attrs_2[-2]
447.    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5
448.
449.
450.    if attrs[-1]==attrs_2[-1]:
451.        distsum=dist+distsum
452.        Nsum=Nsum+1
453.
454.
455.mediaDist=distsum/Nsum
456.distsumkm15=mediaDist/1000
457.print 'Media dist15 en km'
458.print distsumkm15
459.#2016
460.#Agrupar puntos por campos
461.fid = 0   # ID of the feature we will modify
462.
463.layer= QgsMapLayerRegistry.instance().mapLayersByName('ffcc_conex_16_2')[0]
```

```
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm16=mediaDist/1000
print 'Media dist16 en km'
print distsumkm16
mediadist=(distsumkm08+distsumkm09+distsumkm10+distsumkm11+distsumkm12+distsumkm13+dist
sumkm14+distsumkm15+distsumkm16)/9
print 'Media dist en km de ferrocarriles'
print mediadist
```

analiza_pattern_gas.py

```
#2013

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('gas_conex_13')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()



distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm13=mediaDist/1000
print 'Media dist13 en km'
print distsumkm13
#2014

#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('gas_conex_14')[0]
caps = layer.dataProvider().capabilities()
```

```
65.
66. #ponemos 1 en primera fila
67. if caps & QgsVectorDataProvider.ChangeAttributeValues:
68.     attrs = { 0 : 1}
69.     layer.dataProvider().changeAttributeValues({ fid : attrs })
70. layer.updateFields()
71.
72. iter = layer.getFeatures()
73. for feature in iter:
74.     last_id=feature.id()
75.
76.
77.
78. distsum=0
79. Nsum=0
80. counter=-1
81. #Para tomar el ultimo ID
82. iter = layer.getFeatures()
83. for feature in iter:
84.     counter=counter+1
85.     counter_2=counter+1
86.     if counter_2>last_id:
87.         break
88.
89.     fid = counter # the numbered feature (zero based indexing!)
90.     iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
91.     feature = next(iterator)
92.     attrs = feature.attributes()
93.
94.     fid_2 = counter_2 # the numbered feature (zero based indexing!)
95.     iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
96.     feature_2 = next(iterator_2)
97.     attrs_2 = feature_2.attributes()
98.
99.     coordx_1=attrs[-3]
100.     coordy_1=attrs[-2]
101.     coordx_2=attrs_2[-3]
102.     coordy_2=attrs_2[-2]
103.     dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5
104.
105.
106.     if attrs[-1]==attrs_2[-1]:
107.         distsum=dist+distsum
108.         Nsum=Nsum+1
109.
110.
111. mediaDist=distsum/Nsum
112. distsumkm14=mediaDist/1000
113. print 'Media dist14 en km'
114. print distsumkm14
115. #2015
116.
117. #Agrupar puntos por campos
118. fid = 0   # ID of the feature we will modify
119.
120. layer= QgsMapLayerRegistry.instance().mapLayersByName('gas_conex_15')[0]
121. caps = layer.dataProvider().capabilities()
122.
123. #ponemos 1 en primera fila
124. if caps & QgsVectorDataProvider.ChangeAttributeValues:
125.     attrs = { 0 : 1}
126.     layer.dataProvider().changeAttributeValues({ fid : attrs })
127. layer.updateFields()
128.
129. iter = layer.getFeatures()
130. for feature in iter:
```

```
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm15=mediaDist/1000
print 'Media dist15 en km'
print distsumkm15
#2016
#Agrupar puntos por campos
fid = 0   # ID of the feature we will modify

layer= QgsMapLayerRegistry.instance().mapLayersByName('gas_conex_16')[0]
caps = layer.dataProvider().capabilities()

#ponemos 1 en primera fila
if caps & QgsVectorDataProvider.ChangeAttributeValues:
    attrs = { 0 : 1}
    layer.dataProvider().changeAttributeValues({ fid : attrs })
layer.updateFields()

iter = layer.getFeatures()
for feature in iter:
    last_id=feature.id()


distsum=0
Nsum=0
counter=-1
#Para tomar el ultimo ID
iter = layer.getFeatures()
for feature in iter:
    counter=counter+1
```

```
    counter_2=counter+1
    if counter_2>last_id:
        break

    fid = counter # the numbered feature (zero based indexing!)
    iterator = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid))
    feature = next(iterator)
    attrs = feature.attributes()

    fid_2 = counter_2 # the numbered feature (zero based indexing!)
    iterator_2 = layer.getFeatures(QgsFeatureRequest().setFilterFid(fid_2))
    feature_2 = next(iterator_2)
    attrs_2 = feature_2.attributes()

    coordx_1=attrs[-3]
    coordy_1=attrs[-2]
    coordx_2=attrs_2[-3]
    coordy_2=attrs_2[-2]
    dist=((coordx_1-coordx_2)**2+(coordy_1-coordy_2)**2)**0.5


    if attrs[-1]==attrs_2[-1]:
        distsum=dist+distsum
        Nsum=Nsum+1


mediaDist=distsum/Nsum
distsumkm16=mediaDist/1000
print 'Media dist16 en km'
print distsumkm16
mediadist=(distsumkm13+distsumkm14+distsumkm15+distsumkm16)/4
print 'Media dist en km de gaseoductos'
print mediadist
```

REFERENCIAS

[1] F. Sánchez Díaz y C. Torrecillas Lozano, «Diseño de la Red Andaluza de Posicionamiento,» *Mapping,* vol. 94, pp. 6-14, 2004.

[2] Junta de Andalucía, «Portal de Posicionamiento de Andalucía,» [En línea]. Available: www.ideandalucia.es/portal/web/portal-posicionamiento/rap.

[3] V. Ríos, «Correcciones por Post proceso y tiempo real RTK,» 2012.

[4] F. OSGeo, «QGis Web,» [En línea]. Available: https://www.qgis.org/es/site/.

[5] Fundación Python, «Documentación de Python,» [En línea]. Available: https://www.python.org/doc/.

[6] A. C. Müller y S. Guido, Machine Learning with Python, 2016, ISBN: 978-1-449-36941-5.